Clustering markov decision processes for continual transfer

Clustering Markov Decision Processes For Continual Transfer

M. M. Hassan Mahmud[†‡] HMAHMUD42@GMAIL.COM
Majd Hawasly[†] M.HAWASLY@ED.AC.UK
Benjamin Rosman[†⋆+] BROSMAN@CSIR.CO.ZA
Subramanian Ramamoorthy[†] S.RAMAMOORTHY@ED.AC.UK

[†]*School of Informatics*
University of Edinburgh
Edinburgh, EH8 9AB, UK.
‡*Now at: Blue Yonder UK Ltd.*
London, UB11 1FW, UK.
⋆ *Now at: Mobile Intelligent Autonomous Systems Group*
Council for Scientific and Industrial Research
Pretoria, South Africa.
[+] *Now at: School of Computer Science and Applied Mathematics*
University of the Witwatersrand, South Africa

Editor: TBD

Abstract

We present algorithms to effectively represent a set of Markov decision processes (MDPs), whose optimal policies have already been learned, by a smaller *source* subset for lifelong, policy-reuse-based transfer learning in reinforcement learning. This is necessary when the number of *previous* tasks is large and the cost of measuring similarity counteracts the benefit of transfer. The source subset forms an 'ϵ-net' over the original set of MDPs, in the sense that for each previous MDP $\mathcal{M}_p$, there is a source $\mathcal{M}^s$ whose optimal policy has $< \epsilon$ regret in $\mathcal{M}_p$. Our contributions are as follows. We present EXP-3-Transfer, a principled policy-reuse algorithm that *optimally* reuses a given source policy set when learning for a new MDP. We present a framework to cluster the previous MDPs to extract a source subset. The framework consists of (i) a distance d_V over MDPs to measure policy-based similarity between MDPs; (ii) a cost function $g(\cdot)$ that uses d_V to measure how good a particular clustering is for generating useful source tasks for EXP-3-Transfer and (iii) a provably convergent algorithm, MHAV, for finding the optimal clustering. We validate our algorithms through experiments in a surveillance domain.

Keywords: Reinforcement learning, transfer learning, Markov decision process, MDP abstractions, policy reuse, Markov chain Monte Carlo, discrete optimisation.

1. Introduction

Reinforcement learning (RL) in Markov decision processes (MDPs) is a well known framework in machine learning for modelling artificial agents (Puterman, 1994; Sutton and Barto, 1998), where the agent's task is one of sequential decision making. While problems such as policy learning in these MDPs are well posed in terms of an objective such as maximising the expected reward, they are often based on a specific instance of an underlying Markov Decision Process model which may or may not be explicitly known to the learning agent.

Realistic agents must cope with environments with variability, wherein a process generates many instances of MDPs within which the agent must now solve the optimisation problem. Finding optimal policies with respect to an unrestricted family of possible MDPs is not only intractable, but often leads to highly conservative and not practicable solutions. However, many realistic environments actually occupy a middle ground.

In many reinforcement learning problems, that are of great interest in practice, the main source of complexity is partial variability in task description rather than a complete change in the domain. Stated in the language of MDPs, these domains involve a family of possible reward and transition functions over a shared state-action space. This is typically the result of non-stationary behavior of some component that describes the problem. One class of problems is where the human-machine interaction setting remains the same, while the participants change. A recent, highly successful example of such a problem is that of website morphing (Hauser et al., 2009). In this problem, the goal is to present to the user of a website a view/interface adapted to the needs and skill level (savvy, newbie etc.) of that particular user. The interface to present to a user is determined adaptively based on the sequence of her choice of links. So, even though the setting (i.e. the website) remains the same , the dynamics defining the problem (the links the user might choose) changes with each user – i.e. each user corresponds to a new task and corresponding change in problem dynamics. Given the change in dynamics, the algorithm has to determine the best policy/sequence of actions (sequence of interfaces to present). Note that since the algorithm knows that a new user has arrived at the homepage, it always knows if the dynamics may have changed – but not how it has changed. In the MDP formalism, each task corresponds to a particular reward and transition function, while the state and action space remain the same across tasks. Additionally, the reinforcement learning agent is always told when the dynamics/task have changed.

In this paper, we present an approach to dealing with such problems that is based on the notion of transfer learning. We view the new task instance as being similar to previously experienced ones, although we have no explicit measurement of how similar the new task instance is to any previously seen one. Our objective then is to devise an efficient transfer learning method for reinforcement learning in MDPs (Taylor and Stone, 2009) , that enables learning agents to learn efficiently enough to be useful in domains as the one mentioned above.

As a motivating example that we develop further in our experiments, consider a surveillance agent that is monitoring a large geographical region (this is a variant of the kinds of problems that are considered, for instance, in (An et al., 2012)). The agent faces a sequence of monitoring problems where each problem corresponds to the pattern in which infiltrators appear in different locations. If two tasks have similar infiltration patterns, then the same surveillance policy may be good for both of them. During each task, the goal of the agent is to learn the regions where infiltrators appear and choose the appropriate surveillance policy. We do not expect the patterns to be completely different every time, but at the same time we cannot completely rule out a new pattern emerging. In the former case, we should recognize this repetition and take advantage of this fact by reusing old surveillance policies. In the latter case, we should also determine that the new scenario is novel and learn an appropriate policy for that scenario. Furthermore, if the number of previous patterns becomes too large, we also need to compactly re-represent them so that the procedure for determining the correct way to act is more sample efficient.

Our setting of MDPs with changing transition and reward functions are typically referred to as *non-stationary* MDPs and has been considered before (Nilim and Ghaoui, 2005), (Yu et al., 2009) (Yu and Mannor, 2009). In these problems, the goal of the learner is to take actions in such a way

so as to perform as well as the stationary policy which is optimal in hindsight with respect to all the transition and reward function pair the MDP attained. Note that for a specific pair, this policy may be substantially worse than the optimal policy for that pair. Our approach in this paper represents an orthogonal setting, wherein we consider the case where the learner tries to *track* the optimal policy – i.e., always perform as well as the best policy for the current set of transition and reward function – with the proviso that the learner is told when the dynamics have changed. We also require the learner to, if possible, perform better than 'pure' reinforcement learning algorithms applied to specific transition and reward functions by transferring information from the transition and reward functions for which the optimal policy was learned previously.

More formally, in this paper we consider TLRL for the case of a 'lifelong' learning agent that learns a (possibly never-ending) sequence of MDPs which are defined on the same discrete state and action space but differ in terms of the transition and reward distributions. We assume the distribution generating the sequence is unknown and unlearnable (for instance, in the motivating surveillance problem described above, the infiltration pattern may depend on the internal variables of the infiltrators that are not known to us). In this setting, the goal of the agent is to, if possible, *reuse* the optimal policies in the previous MDPs in order to learn the new MDP more *efficiently*. In this continual setting, we assume that the agent operates in the new MDP for a fixed number of episodes, and hence we measure efficiency by the total reward accumulated while learning the new task during these fixed number of episodes. Reusing a policy means that we should try the optimal policies of the previous MDPs in the new MDP and if one results in efficient behavior we should keep using it. However, as we described above, a problem in this setting is that, when the number of previous tasks become too large, transfer becomes ineffective as the agent spends too much time testing the old policies. In this instance, one possible solution to this problem is to find a subset of the N *previous* policies, which we call *source* policies, that are, in a well-defined and useful sense, representative of all the N previous policies (see Section 1.1 for alternative encodings). In other words, the source policies should form the analogue of an ϵ-net in a metric space (Kolmogorov and Fomin, 1970) over the space of previous MDPs with respect to an appropriate distance over MDPs. In this paper we present a clustering based approach to finding this smaller subset of source policies.

Our overall approach is illustrated in figure 1. The main idea is to cluster the N previous MDPs into c clusters, where the number c and the clusters themselves are to be determined via discrete optimization, and then choose the representative element of each cluster to obtain the source MDPs. The optimal policies of the source MDPs then become the source policies. In our approach to choosing the clustering and the corresponding source policies, we attempt to ensure a-priori that the chosen source policies are a good representative of the previous tasks for the purposes of transferring to the unknown target task.

In particular, we define a transfer learning algorithm, EXP-3-Transfer, with performance bound $g(c)$ that depends on the number c of source policies. Hence this explicitly measures how good the size of the clustering c is. We are now left with the task of choosing the clustering and the corresponding source policies. To that end, we define a distance function d_V between two MDPs that takes the distance between two MDPs to be how well the optimal policy of one MDP performs in the other (this in turn is measured as the difference between the value of the optimal policy of the former and the value of the optimal policy of the latter at the start states of the latter). Hence, given that our goal is to reuse optimal policies of one MDP in another, we choose our clustering so that within each cluster the pairwise d_V distances between the elements of the cluster are low. Similarly, we choose the source policy for each cluster to be the optimal policy of the MDP in that cluster

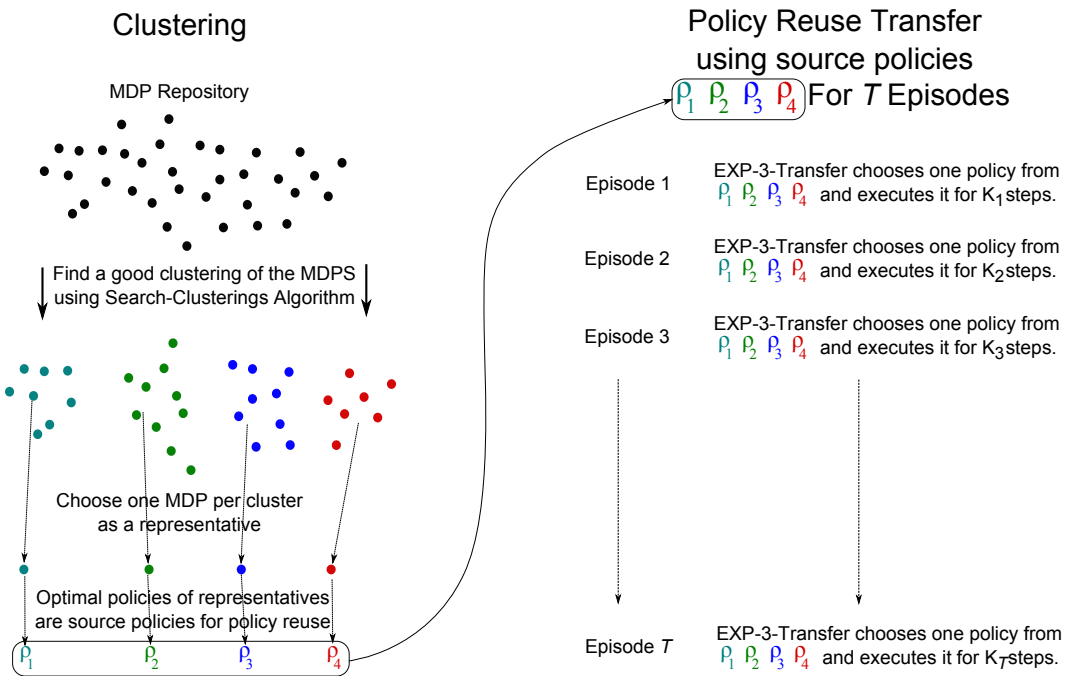


Figure 1: This figure illustrates our overall algorithmic approach. Our method consists of two parts. In the first part (left pane) we use the Search-Clusterings algorithm to find a good clustering of the MDPs in our repository, which are then used to generate a source set of policies. In the second part (right pane) these policies are used by EXP-3-Transfer to perform transfer using policy reuse in a new MDP. Reducing the number of policies input to EXP-3-Transfer helps it choose the best policy quickly. On the other hand, reducing policies risks leaving out a policy with good performance in the new MDP. The bulk of the paper is dedicated to deriving a principled way to trade off these two contradictory goals .

which has low d_V distance with respect to all the other elements. Hence, the cost of a clustering with c clusters is, roughly speaking, $g(c) + \epsilon$ where ϵ is a measure of the inter-element d_V distances in the clusters.

Given the cost function, we show that it is NP-hard to find the optimal clustering. So, we introduce a Markov chain Monte Carlo based discrete optimization algorithm to find it. The algorithm is an extension of the Metropolis-Hastings algorithm, which we call Metropolis-Hastings with Auxiliary Variables (MHAV in short), and can also be thought of as an extension to simulated annealing (Kirkpatrick et al., 1983) with stochastic temperature changes. Simulated annealing is a well known algorithm for discrete optimization, but requires carefully setting up an infinite sequence of parameters known as the temperature schedule. Determining this schedule in practice to ensure convergence is acknowledged to be very difficult, and practically an art form. In our version of the algorithm, we search over both the temperature and the optimal point simultaneously, thereby handling the problem of setting the schedule automatically.

To summarize, our overall continual transfer algorithm is as follows. The agent continually learns optimal policies for MDPs presented in sequence. When learning the optimal policy for a particular MDP, the learning agent uses the optimal policies of the previously solved MDPs in a policy reuse transfer learning algorithm. To make transfer more effective, at fixed intervals, the agent clusters the previous MDPs to derive a small subset as the set of source MDPs, whose optimal policies are then used as input to the policy reuse algorithm. The clustering is chosen so as to optimize the regret of the transfer algorithm, and is found by using a convergent discrete optimization algorithm.

We conclude this brief introduction to our method by noting that our transfer algorithm EXP-3-Transfer is in fact an extension of the well known EXP-3 algorithm (Auer et al., 2002b) for non-stochastic multi-armed bandits, and our performance bound $g(c)$ is in fact a regret bound of the type well known in bandit algorithm literature. Our strategy is to cast the policy reuse transfer learning problem as a bandit problem, with 'pure reinforcement learning algorithm' as one arm, and the c source policies as the remaining arms. The regret bound for EXP-3 ensures that we minimize negative transfer by never performing much worse than pure reinforcement learning. We will now discuss related work.

1.1 Related Work

As evidenced by the survey paper (Taylor and Stone, 2009), a significant amount of work has been done on transfer learning in reinforcement learning. As mentioned previously, lifelong learning in reinforcement learning was first explicitly considered in (Mitchell and Thrun, 1993; Thrun and Mitchell, 1995; Thrun, 1996) in the context of learning in robots. In these works, the main aim was to learn the dynamics of robot motion in one circumstance using a function approximator (such as neural networks) and then use these learnt dynamics as an initial bias in a new situation using an explanation based learning framework.

In terms of recent work on TLRL, two different strands are related to our work. The first is work on policy reuse and the second on task encoding. The first and, to the best of our knowledge, the only authors to have considered policy reuse are (Fernandez et al., 2006, 2010). The algorithms presented there, at the beginning of every episode, choose between different source policies by using a softmax criteria on accumulated reward and then use the chosen policy as an initial exploration policy before switching to Q-learning exclusively. In contrast, we extend the EXP-3 algorithm for

multi-armed bandits to choose between source policies and Q-learning, and as a result inherit its theoretical guarantees. Additionally, they do not consider the problem of source task selection, whereas in our work this is a major focus.

Two closely related works are (Talvitie and Singh, 2007) and (Azar et al., 2013) where the authors consider the best way to choose between a set of stationary policies. The main distinction between policy reuse and these is the requirement in the former to not perform much worse than a base RL algorithm (which is also our requirement).

We now look at previous work that uses a smaller set of source tasks to represent the complete set of previous tasks. The problem of source task selection through clustering seems to have been considered only by Carroll and Seppi (2005). They introduce several measures for task similarity and then consider clustering tasks according to those measures. The distance functions introduced were heuristic, and the clustering algorithm used was a simple greedy approach. The evaluation of their method was on several toy problems. In contrast, we derive a cost function for clustering in a principled way to optimize the regret of our EXP-3-Transfer policy reuse algorithm. Additionally, instead of constructing the cluster in a greedy fashion, we search through clustering space using a convergent discrete optimization algorithm.

Recent work that also chooses selectively from previous tasks is (Lazaric et al., 2008), (Lazaric and Restilli, 2011). The setting for this paper is a collection of tasks defined on the same state-action space with the tasks and the state-action-state triples for the different tasks generated i.i.d. (similar to the multi-task transfer in classification setting considered in (Baxter, 2000)) rather than sequentially as is typical in RL. Under this setting the authors are able to bound the error when samples from one task are used to learn the new task. This is quite a different setting from us as it is 'batch' RL rather than the typical online and sequential RL and measures similarity in terms of the actual transition and reward functions rather than policies or values. Additionally, the analysis and algorithms are derived under the assumption of a fixed set of prior tasks rather than the continual lifelong learning setting we consider.

Source task selection is not the only possible way to represent previous tasks, and the overall goal of finding abstractions for exploiting commonality has received considerable attention in the transfer learning community. Most of the work done in deriving abstractions for the purposes of transfer has been for MDP homomorphisms (Ravindran and Barto, 2003; Ferns et al., 2004; Ravindran, 2013; Konidaris and Barto, 2007; Sorg and Singh, 2009; Castro and Precup, 2010). In these works, similarity between MDPs is defined in terms of bisimulation between states of different MDPs. Bisimulation is a concept borrowed from process algebra. In the context of transfer learning in MDPs, at its most general formulation, a bisimulation is an isomorphism f, g between the state and action spaces that is preserved under the transition distribution – that is for every state-action-state triple s, a, s', $T_1(s'|s, a) = T_2(f(s')|f(s), g(a))$ where T_i are the transition distribution of the two MDPs. Unfortunately, in this pure form, bisimulation is absolute (two MDPs are either bisimilar or not) and does not take into account the reward function. And so, in the papers mentioned above, this basic notion was extended in various ways to address both these issues. However, one of the main issues with bisimulation is computational cost, and this remains so in the extensions as well. Another issue with these approaches is that, as observed by Castro and Precup (2010), bisimulation is a worst case metric (two states may have identical optimal actions but still be completely different according to the metric) and as a result requires heuristic modifications.

Technically, the main difference between our approach and bisimulation based methods is that the similarity between different MDPs are ultimately determined by distance between value func-

tions. In our case, however, we are interested in distance in terms of *policy*. As a result, even though two tasks might be quite different in terms of the value function they might be identical in terms of the optimal policy, and our approach will capture this (as noted earlier, failing to do this was one of the issues with bisimulation based approaches).

Another interesting line of work that uses a different approach to abstracting MDPs is the proto-value function based approach of (Ferrante et al., 2008). Proto-value functions were introduced in (Mahadevan, 2005) as an efficient way to represent the value function for large state spaces as a linear combination of functions, which are called proto-value functions. The main innovation in this approach is that, in representing the value function as a real function over state space, the state-space is treated as a manifold where the distance between points/states is determined by the reachability graph of the MDP. This idea of a spectral-decomposition of the value function naturally lends itself to transfer learning, as, given a new task, we can imagine using the proto-value functions learned in a previous task to initialize the new value function in the new task. It has been noted that proto-value function based transfer has issues in terms of scalability and tractability. The main difference between this and our work is that, as with the homomorphism based approach, our similarity notion is based on policy similarity, while theirs is based on similarity between value functions. Identifying policy similarity is more desirable because tasks similar in terms of value function will be similar in terms of policy, but not necessarily the other way round.

A somewhat different approach to finding MDP abstractions was adopted in (Ammar et al., 2012), where MDPs were related by mapping state action state triples to a lower dimensional space, and consider triples to be equivalent if their representations were found to be close. The authors were able to transfer between tasks such as inverted pendulum, cart pole and mountain car, showing that in these cases their approach was able to discover the fact that the differential equations describing these domains have similar/identical forms.

Finally, our work can also be related to the notion of equivalence between probabilistic models, which has been influential in early work on Bayesian network learning. For instance, Chickering and collaborators wrote a series of papers in which the notion of event equivalence and score equivalence is used to render the problem of searching over network structures somewhat tractable. In (Heckerman and Chickering, 1995), it is shown that the notion of event equivalence, i.e., that two Bayesian networks should be treated as similar if they agree on the independence and dependence relationships between random variables, can be used to define local structure edit operations that enable learning of network structures. Subsequently, in (Chickering and Boutilier, 2002), (Chickering, 2003) this idea is developed further to show that by considering this notion of equivalence it is possible to achieve optimal structure identification with an essentially greedy procedure. We take inspiration from such work, but also note that our task of comparing two sequential decision making problems differs from that of making predictions with a probabilistic model, calling for new notions of process similarity and corresponding algorithms for transfer.

1.2 Paper Organization

In the following we proceed as follows. We present the notation and some fundamental notions in Section 2. Then we define our transfer learning algorithm and framework for measuring distance in Section 4 respectively. Sections 5 and 6 presents our clustering algorithm and the full continual transfer algorithm. We then present our experiments in Section 7 and then end with a conclusion in Section 8.

2. Preliminaries

We use $\triangleq$ for definitions, Pr to denote probability and $\mathbb{E}$ for expectation. A finite MDP $\mathcal{M}$ is defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, P, R, \gamma)$ where $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ is a finite set of actions and $\mathcal{R} = [l, u] \subset \mathbb{R}$ is the set of rewards. $P(s'|s, a)$ is a the state transition distribution for $s, s' \in \mathcal{S}$ and $a \in \mathcal{A}$ while $R(s, a)$, the reward function, is a random variable taking values in $\mathcal{R}$. Finally, $\gamma \in [0, 1)$ is the discount rate.

A (stationary) policy π for $\mathcal{M}$ is a map $\pi : \mathcal{S} \to \mathcal{A}$. For a policy π, the Q function $Q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is given by $Q^\pi(s, a) = \mathbb{E}[R(s, a)] + \gamma \sum_{s'} P(s'|s, a)Q(s, \pi(s'))$. The value function for π is defined as $V^\pi(s) = Q^\pi(s, \pi(s))$. An optimal policy π^* satisfies $V^{\pi^*}(s) \geq V^\pi(s)$ for all policy π and state s – it can be shown that every finite MDP has an optimal policy. V^{π^*} is written V^*, and the corresponding Q function is given by $Q^*(s, a) = \mathbb{E}[R(s, a)] + \gamma \sum_{s'} P(s'|s, a)Q^*(s', \pi^*(s'))$. When the agent acts in the MDP, at each step it takes an action a at a state s, and moves to the next state s' and the reward r. The goal of the agent is to learn π^* from these observations and then choose the action $\pi^*(s)$ at each state. If there are multiple optimal policies, we will designate the first policy in a lexicographical order as the canonical policy. We will assume that $R_{\max}$ is a known upper bound on the reward function. Without loss of generality, in the sequel we assume that there is a single initial state s°. We call a policy π ϵ-*optimal* if $V^*(s^\circ) - V^\pi(s^\circ) \leq \epsilon$

The transfer learning setting. In the transfer learning setting, we are given previous MDPs $\mathcal{M}_i$, $1 \leq i \leq N$ and we transfer from these tasks to learn the $N + 1^{st}$ MDP $\mathcal{M}_{N+1}$. The learning of $\mathcal{M}_{N+1}$ runs for T episodes. We denote the optimal policy of the i^{th} previous MDP by π_i^*, and the value of a policy π in MDP i as V_i^π. Similarly, we denote the reward and transition functions of the i^{th} MDP by P_i and R_i respectively. We will assume that the rewards of all MDPs fall within the range $[R_{\min}, R_{\max}]$ and we define $\Delta R \triangleq R_{\max} - R_{\min}$.

3. The Policy Reuse Problem

In this section we first concretely define the problem of policy-reuse for transfer learning and then define our algorithm for solving this problem. We define the goal of policy reuse to be to design an algorithm that runs for T episodes on the target task $\mathcal{M}_{N+1}$ and, given a collection of *source policies* $\rho_1, \rho_2, \cdots, \rho_c$ and the Q-learning algorithm/policy, performs nearly as well as the best policy (in hindsight) in this collection over the T episodes. This requirement has two important implications. First, since the set of policies contain Q-learning, and as Q-learning converges to the optimal policy, it means the algorithm is required to perform *nearly as well as a policy that converges to the optimal policy* – in other words, the algorithm should *avoid negative transfer*. Second, if there is a source policy that is near-optimal, then the algorithm is also required to perform *nearly as well as that near-optimal policy* – in other words, the algorithm should *transfer from the new to the old task* if possible.

To derive our algorithm we show that policy reuse may be cast as an instance of the non-stochastic multi-armed bandits problem, and hence the classic EXP-3 algorithm (Auer et al., 2002b) may be extended to solve policy reuse. We call our extension EXP-3-Transfer and we derive regret bounds for this algorithm (which is similar to the bounds for EXP-3). In particular we show that EXP-3-Transfer performs nearly as well as the best policy in the collection described above, in expectation with respect to the randomness in the algorithm and the reward and transition function of

$\mathcal{M}_{N+1}$. Hence, this algorithm satisfies the requirements we laid out for a policy reuse algorithm. We now expand these ideas.

3.1 Defining The Policy Reuse Problem

In the policy reuse transfer problem, we have the target task $\mathcal{M}_{N+1}$ and a set of c source policies $\rho_1, \rho_2, \cdots, \rho_c$ (in general, ρ_is are arbitrary – but in this paper, each ρ_i is the optimal policy π_j^* of some $\mathcal{M}_j$ our repository, and Section 4 shows how to choose the ρ_is in a principled way). It seems that policy reuse as a method for transfer was first introduced in (Fernandez et al., 2006, 2010). The algorithm introduced was called Probabilistic Policy Reuse (PPR), and the goal of the algorithm was to balance using the c source policies and pure reinforcement learning policy (ϵ-greedy Q-learning) so that the learning algorithm converges faster than running pure RL by itself. The basic idea in PPR is as follows. At the beginning of each episode, PPR chooses a policy from among the source policies and the ϵ-greedy Q-learning policy using a softmax criterion on the observed returns of the policies in previous episodes. It then initiates a policy-reuse episode, where at each step it probabilistically chooses between ϵ-greedy Q-learning and the chosen policy, with probability of choosing ϵ-greedy Q-learning going to 1 during the episode. In essence, the c source policies serve as an initial exploration policy, so that if they to take the agent through paths of the optimal policy, it would result in faster learning of the optimal policy.

There are several aspects of the above algorithm that are noteworthy. First, even if the c source policies contain the optimal policy, the algorithm would deterministically switch to Q-learning after the initial phase. Another aspect is that, while there is an intuitive connection between the soft-max criterion and the benefit of using a policy, the actual connection is not made rigorous. Both these issues arise from the fact that the goal of policy reuse was not defined concretely in (Fernandez et al., 2006). So, taking cue from the definition of online learning algorithms (Vovk, 1990; Littlestone and Warmuth, 1994; Cesa-Bianchi and Lugosi, 2006), we define the policy reuse problem concretely as designing an algorithm that chooses policies at every episode in such a way that it does not perform much worse than any of the c policies or Q-learning over the T episodes. As discussed above, this requirement implies both that (i) the algorithm avoids negative transfer and (ii) transfers from/reuses a good policy (if one exists). Formally,

Definition 1 (The policy reuse problem) *Let transfer learning for the target task $\mathcal{M}_{N+1}$ be run for T episodes, and let $\bar{x}_i(t) \triangleq \sum_{n=1}^{K_t} \gamma^n r_n$ be the discounted return accumulated by running ρ_i (with ρ_{c+1} being the non-stationary Q-learning policy) at episode t, with r_n the reward at step n and K_t the length of the episode. Let the total discounted reward for policy ρ_i be $R_i(T) = \sum_{t=1}^{T} \bar{x}_i(t)$. Let $R_A(T)$ be the total discounted reward accumulated by a policy-reuse algorithm A. Then we require that for each i*

$$\mathbb{E}_{R_{N+1},P_{N+1}}\left[\mathbb{E}_A[R_i(T) - R_A(T)]\right] = o(T)$$

where $\mathbb{E}_{R_{N+1},P_{N+1}}$ is the expectation is with respect the dynamics of $\mathcal{M}_{N+1}$ and $\mathbb{E}_A$ is the expectation with respect to randomization in the algorithm.

In the following subsections we present the EXP-3-Transfer algorithm for solving this problem and then analyze its performance.

3.2 EXP-3-Transfer For Policy Reuse

In this section, we introduce the non-stochastic multi-armed bandits problem (Auer et al., 2002b) and show that the policy reuse problem may be cast as in instance of this problem. We then present EXP-3-Transfer to solve this problem, which is an extension/modification of the classic EXP-3 algorithm (Auer et al., 2002b) for non-stochastic multi-armed bandits. We discuss the difference between EXP-3 and EXP-3-Transfer in Section 3.3.

NON-STOCHASTIC MULTI-ARMED BANDITS

The non-stochastic multi-armed bandits problem is defined as follows.

- There are $c+1$ *arms* where each arm i has a *payoff process* $x_i(t)$ associated with it. No assumptions are made on the payoff process $x_i(t)$ and they may in fact be adversarial (this is the reason the problem is called non-stochastic).
- A learner operates for T steps and at each step t it pulls/selects one of the arms i_t.
- At step t, the learner obtains payoff $x_{i_t}(t)$ and only observes the payoff of the arm i_t it has selected.

The goal of the learner is to minimize its *regret* with respect to the best arm, that is choose arms $i_1, i_2, \cdots, i_T$ to minimize the quantity

$$Reg(i_1, i_2, \cdots, i_T) \triangleq \max_i \sum_{t=1}^{T} x_i(t) - \sum_{t=1}^{T} x_{i_t}(t)$$

A randomized algorithm, called EXP-3, was developed in (Auer et al., 2002b) which minimizes the *expected regret*, where the expectation is taken with respect to the randomization in the algorithm. It turns out that the regret of EXP-3 satisfies the requirements for the regret R_A in Definition 1 (we expand on this in the next subsection). This implies that the non-stochastic bandits approach, extended to our setting, solves the problem of policy reuse.

POLICY REUSE AS NON-STOCASTIC MULTI-ARMED BANDITS

We may cast the policy reuse problem as a non-stochastic bandit problem as follows.

- Each source policy ρ_i and the the Q-learning policy corresponds to an arm, giving a total of $c+1$ arms.
- At each step t in the bandit problem, we (the learner) select a policy/arm ρ_{i_t}, and execute it for an *episode* in the target $\mathcal{M}_{N+1}$ (so a step in the multi-armed bandit problem corresponds to an episode in the policy reuse problem).
- The payoff we receive for executing policy ρ_{i_t} (i.e. choosing arm i_t) is the (normalized) total discounted reward by executing ρ_{i_t} for that episode: $x_{i_t}(t) = [\Delta R(1-\gamma)^{-1}][\sum_{n=1}^{K_t} \gamma^n r_n - R_{\min}(1-\gamma)^{-1}]$, with r_n the reward at step n and K_t the length of the episode (the normalization is required as EXP-3 expects arm payoffs to lie in $[0,1]$).

EXP-3-TRANSFER

Given the above transformation, our algorithm EXP-3-Transfer for choosing the policy i_t at step t is given in pseudocode form in Algorithm 1. The basic idea is straightforward – at each step it chooses a policy with probability proportional to an *adjusted* cumulative observed reward of the policy and a term to encourage exploring the policy if it has not been selected recently. The rewards are adjusted to compensate for the fact that, at each step t, the algorithm observes the payoff of only ρ_{i_t}, the policy chosen at that step (see Section 3.3 for details on the adjustment). In addition, if the algorithm determines that with high probability a particular source policy (i.e. all but the Q-learning policy) is worse than some other source policy, then it eliminates it from further consideration. The idea is that if we remove an arm we know to be sub-optimal, then we save the episodes that would have been wasted trying that policy.

In detail, the main loop of EXP-3-Transfer runs from line 3 to line 18. C_t contains all the policies not yet eliminated by EXP-3-Transfer. Line 4 computes the probability of choosing a policy, which is proportional to the adjusted observed payoff of each policy, plus the exploration term ($\beta/|C_t|$). In the next two steps, a policy is chosen probabilistically, executed and its payoff observed and normalized (normalized because EXP-3 expects payoffs to lie in $[0, 1]$). In line 7 we record the payoff of the executed policy for use in the elimination step. In lines 8 and 9, the adjusted payoffs of the policies are computed, and their weights updated, respectively. Finally, steps 11 to 17 looks at each stationary policy in C_t, and checks to see if average *non-adjusted* payoff so far satisfies the elimination condition. If so, the policy is removed from C_t. The elimination condition is justified/obtained from the theorems below which derive the regret bound of the algorithm.

3.3 Analysis of EXP-3-Transfer

To begin analysis of EXP-3-Transfer, we first note that the payoffs/discounted cumulative reward $\bar{x}_i(t)$ of a source policy ρ_i is i.i.d., while the payoff of the Q-learning arm has an unknown non-stationary distribution (because the choice of actions in Q-learning is non-stationary). Our strategy for analyzing the performance despite the non-stationarity, is to assume that there is an unknown adversary that is generating the payoffs for Q-learning and then bound the expected worst-case regret of EXP-3-Transfer with respect to this adversary (this is *exactly* the strategy used to analyze EXP-3). In particular, in our adversarial/worst-case analysis we assume that there are three participants Nature, Adversary and Player, who make the following choices, in that order.

1. Player chooses the number of episodes T and the source policies $\rho_1, \rho_2, \cdots, \rho_c$, and all the other parameters for EXP-3-Transfer
2. Adversary chooses the payoffs for T episodes of Q-learning
3. Nature chooses the episodic payoffs $\bar{x}_i(t)$ for each of the policies ρ_i for $1 \leq t \leq T$, according the i.i.d. distribution that governs the payoffs (the distribution is determined by P_{N+1}, R_{N+1}, the transition and reward functions respectively of $\mathcal{M}_{N+1}$, and ρ_i).
4. Player now runs EXP-3-Transfer for T episodes, and observed payoffs x_{i_t} are the ones chosen by Nature and Adversary as above.

The important thing to note is that in the above framework, there are two sources of randomness/stochasticity – first in step 3 due to the randomization due to the transition and reward functions

Algorithm 1 EXP-3-Transfer($\mathcal{M}, \{\rho_1, \rho_2, \cdots, \rho_c\}, \beta, T, l, \Delta R, \delta$)

1: **Input:** MDP $\mathcal{M}$, policies 1 to c: the source policies $\rho_1, \cdots, \rho_c$ and EXP-3 parameters β and T; l the interval at which to eliminate policies; ΔR, upper bound on the range of the per step rewards; δ, confidence parameter for eliminating source policies.

2: **Initialize:**

- Set Q-learning policy as the $c + 1^{th}$ policy.
- Set $w_i(1) = 1$, let $x_i(t)$ be the payoff of the policy i at step t; let $C_1 \triangleq \{\rho_1, \rho_2, \cdots, \rho_c, \text{Q-learning policy}\}$.
- Set $n_i \leftarrow 0$, where $1 \leq i \leq c$, and n_i is the number of times ρ_i has been used; set $z_i \leftarrow 0$, where $1 \leq i \leq c$, and z_i is the total normalized discounted reward observed for ρ_i when it was executed.

3: **for** $t = 1$ to T **do**

4: **If** $\rho_i \in C_t$ **then** set $p_i(t) = (1-\beta)\frac{w_i(t)}{\sum_{\rho_i \in C_t} w_i(t)} + \frac{\beta}{|C_t|}$; else set $p_i(t) = 0$.

5: Select policy ρ_{i_t} for step t to be i with probability $p_i(t)$, increment $n_{i_t} \leftarrow n_{i_t} + 1$.

6: Execute policy ρ_{i_t} for one-episode, and observe discounted payoff $\bar{x}_{i_t}(t)$; normalize $x_{i_t}(t) \leftarrow [\bar{x}_{i_t}(t) - R_{\min}(1-\gamma)^{-1}]/[\Delta R(1-\gamma)^{-1}]$.

7: **if** ρ_{i_t} is not the Q-learning policy **then** set $z_{i_t} \leftarrow z_{i_t} + x_{i_t}(t)$.

8: For each $\rho_i \in C_t$, set

$$\hat{x}_i(t) \leftarrow \begin{cases} x_i(t)/p_i(t) & \text{if } i = i_t \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

9: For each $\rho_i \in C_t$, set $w_i(t+1) \leftarrow w_i(t) \exp[\beta \hat{x}_i(t)/(c+1)]$.

10: Set $C_{t+1} \leftarrow C_t$.

11: **if** $t \mod l = 0$ **then**

12: **for** $k = 1$ to c, $\rho_k \in C_t$ **do**

13: **if** $\exists \rho_j \in C_t$, $j \leq c$, such that, for $\epsilon = z_j/n_j - z_k/n_k$, we have $\epsilon/2 > \sqrt{-\ln(\delta/2c)(2n_j)^{-1}}$ and $\epsilon/2 > \sqrt{-\ln(\delta/2c)(2n_k)^{-1}}$ **then**

14: Set $C_{t+1} \leftarrow C_t - \{\rho_k\}$.

15: **end if**

16: **end for**

17: **end if**

18: **end for**

of $\mathcal{M}_{N+1}$ and second in step 4 due to choices made by EXP-3-Transfer. As such, our regret bound holds in expectation with respect to these two sources, which we denote by $\mathbb{E}_{R,P}$ and $\mathbb{E}_{E3T}$, respectively. The former is expectation with respect to the transition and reward function in $\mathcal{M}_{N+1}$, while the latter is expectation with respect to EXP-3-Transfer. It is crucial to note that in taking these expectations, we assume that the choices made by the adversary in step 2 are taken to be fixed (this is identical to what was done in the analysis of EXP-3 in Auer et al. (2002a)). Our main result for this section is as follows:

Theorem 2 *For any source policy $\rho_j \in C_T$,*

$$V_{N+1}^{\rho_j} - \frac{1}{T}\mathbb{E}_{R_{N+1},P_{N+1}}\left[\mathbb{E}_{E3T}\left[\sum_{t=1}^{T}\bar{x}_{i_t}(t)\middle|\rho_j \in C_T\right]\right] \leq \frac{\Delta R}{1-\gamma}\left(\frac{|C_T|}{\sqrt{c+1}} + \sqrt{c+1}\right)\times \sqrt{(e-1)\ln(c+1)/T} \quad (2)$$

$$\leq \frac{\Delta R}{1-\gamma}2\sqrt{e-1}\sqrt{(c+1)\ln(c+1)/T}$$

$$= \frac{\Delta R}{1-\gamma}2.63\sqrt{(c+1)\ln(c+1)/T} \quad (3)$$

when the algorithm is run with $\beta = \min\{1, \sqrt{(c+1)\ln(c+1)/[T(e-1)]}\}$. Here $\mathbb{E}_{R_{N+1},P_{N+1}}$ is expectation with respect to the distributions R_{N+1} and P_{N+1} of $\mathcal{M}_{N+1}$, while $\mathbb{E}_{E3T}[\cdot|\rho_j \in C_T]$ is the expectation with respect to randomization of EXP-3-Transfer, conditional on the event that $\rho_j \in C_T$.

Additionally, with probability $1-\delta$, with respect to randomization due to the target MDP $\mathcal{M}$, for all source policy $\rho_i \notin C_T$, there is at least one $\rho_{i'} \in C_T$ such that

$$V_{N+1}^{\rho_i} < V_{N+1}^{\rho_{i'}}$$

The proof is given in Appendix A, but in the following we discuss the bound itself. For the sequel, we define the following function:

$$g(c,T) \triangleq \Delta R(1-\gamma)^{-1}2.63\sqrt{(c+1)\ln(c+1)/T} \quad (4)$$

This is the right side of the bound above (we will also use it in Section 4 to quantify how good a particular set of source policies is for the purpose of transfer).

The first issue we need to check is whether the bound satisfies the requirements we set out in Definition 1. The definition requires that $\mathbb{E}_{R_{N+1},P_{N+1}}\mathbb{E}_A[R_i(T) - R_A(T)] = o(T)$ where $R_i(T)$ and $R_A(T)$ are the total discounted reward over T episodes and the expectations are with respect the randomization due to reward and transition function of $\mathcal{M}_{N+1}$ and the randomization in the algorithm (i.e. $\mathbb{E}_A \equiv \mathbb{E}_{E3T}$).

We consider the edge case when $\delta = 0$, and so no arms are ever eliminated and all $\rho_j \in C_T$. In this case, for any ρ_j,

$$\mathbb{E}_{R_{N+1},P_{N+1}}[\mathbb{E}_A[R_j(T)]] = \mathbb{E}_{R_{N+1},P_{N+1}}[R_j(T)] = T \times V_{N+1}^{\rho_j}, \quad (5)$$

which is T times the first term in the left hand side of the bound. Now, for any ρ_j

$$\mathbb{E}_{R_{N+1},P_{N+1}}\mathbb{E}_A[R_A(T)] = \mathbb{E}_{R_{N+1},P_{N+1}}\left(P_{E3T}(\rho_j \in C_T)\mathbb{E}_{E3T}\left[\sum_{t=1}^{T}\bar{x}_{i_t}(t)\middle|\rho_j \in C_T\right] + P_{E3T}(\rho_j \notin C_T)\mathbb{E}_{E3T}\left[\sum_{t=1}^{T}\bar{x}_{i_t}(t)\middle|\rho_j \notin C_T\right]\right)$$

$$= \mathbb{E}_{R_{N+1},P_{N+1}}\mathbb{E}_{E3T}\left[\sum_{t=1}^{T}\bar{x}_{i_t}(t)\middle|\rho_j \in C_T\right] \quad (6)$$

By the first part of Theorem 2,

$$V_{N+1}^{\rho_j} - \mathbb{E}_{R_{N+1},P_{N+1}}\mathbb{E}_A[R_A(T)] \leq g(c,T) = o(T)$$

which is what is required. Now recall from discussion of Definition 1 above that satisfying the regret requirements imply that the algorithm avoids negative transfer and transfers when possible. Hence, the theorem shows that EXP-3-Transfer does indeed avoid negative transfer and transfers when possible.

When we move δ away from 0, we trade off adherence to the requirement in Definition 1 for practical performance. Indeed, while conducting the experiments reported in Section 7, we observed that setting δ to a suitably low but non-zero value greatly improves performance. However theoretical guarantees are not completely lost in this instance, because for the source policies that are eliminated, part 2 of the theorem says that there is at least one source policy which is, with high probability strictly better than the other policy.

We end this section with a brief discussion of the difference between EXP-3 and EXP-3-Transfer. The primary difference is that we eliminate policies/arms to improve practical performance of the algorithm. This is not possible in EXP-3 because, unlike in our case, it is not assumed that some of the arms have i.i.d. payoffs and so their means cannot be estimated from observations. This leads to a slightly different analysis and improved constant $\frac{|C_T|}{\sqrt{c+1}} + \sqrt{c+1}$ for EXP-3-Transfer and 2 for EXP-3. The practical difference we observe was actually considerable, and without it we were not able to outperform PPR in our experiments (see below).

4. The Clustering Approach To Task Encoding

Recall from Section 1 that in our problem, we assume that we have a repository of MDPs and their optimal policies and we wish to use the optimal policies as the source policies for EXP-3-Transfer. Now we have the following dilemma. As the number of source policies increase, we spend more time evaluating them and accruing sub-optimal reward in the process. On the other hand, if we choose a subset of policies from the repository, then we risk leaving out policies that may have been very useful in the new task. So, essentially we are faced with the problem trading off the size and diversity of the set of source policies.

In this section, we concretely define this tradeoff problem as optimizing a cost function, and in Section 5 we describe an algorithm to optimize it. The cost function is defined over the set of *clusterings*/partitions of the repository, where each clustering is used to choose a particular subset of policies in the repository as the source policy set. We first show how to choose this source set given a clustering, and then define the cost function that measures how well the source set achieves the tradeoff mentioned above. This cost function then helps us choose the optimal clustering and hence the optimal source policies.

4.1 Constructing Source Policies Given Clustering

Given the N MDPs $\mathcal{M}_1, \mathcal{M}_2 \cdots, \mathcal{M}_N$ in our repository, let $\{A_1, A_2, \cdots, A_c\} \triangleq \mathbf{A}$ be a particular clustering – that is $\cup_i A_i = \{\mathcal{M}_1, \mathcal{M}_2 \cdots, \mathcal{M}_N\}$ and $A_i \cap A_j = \emptyset$. The set of clusterings may vary both by elements of each A_i, and the number c of *cells* A_i. Given a particular clustering $\mathbf{A}$, we will obtain the c source policies by choosing one policy per A_i. To choose this policy, we define a distance function that measures how similar two MDPs are in terms of their optimal policies.

Let $\mathcal{M}_i$ and $\mathcal{M}_j$ be any two MDPs in our repository (the distance definition generalizes to any two MDPs defined on the same state and action space, but with different transition and reward functions). Recall that V_i^π and V_j^π denotes the value of policy π when executed, respectively, in $\mathcal{M}_i$ and $\mathcal{M}_j$ at the initial state (this can be generalized to different initial states and/or distribution over initial states very easily – but we consider the same initial state setting to keep the presentation simple). Letting the optimal policies for the two MDPs be π_i^*, π_j^*, we define the *optimal policy similarity* between two MDPs as follows.

$$d_V(\mathcal{M}_i, \mathcal{M}_j) \triangleq \max\{V_i^* - V_i^{\pi_j^*}, V_j^* - V_j^{\pi_i^*}\} \tag{7}$$

So this distance upper bounds how much we lose if we use the optimal policy of one MDP in the other – in particular we have the following lemma by construction.

Lemma 3 *If $d_V(\mathcal{M}_i, \mathcal{M}_j) \leq \epsilon$, then the optimal policy of $\mathcal{M}_i$ is at least ϵ-optimal in $\mathcal{M}_j$ and vice versa.*

The definition of d_V is motivated by the fact that the goal of policy reuse is to use the optimal policy of one MDP in another. We now define the source policies given a clustering $\mathbf{A}$:

Definition 4 *Given a clustering $\mathbf{A} = \{A_1, A_2, \cdots, A_c\}$, define for each A_i the MDP $\mathcal{M}^i$ as follows:*

$$\mathcal{M}^i \triangleq \arg\min_{\mathcal{M} \in A_i} \max_{\mathcal{M}' \in A_i} d_V(\mathcal{M}, \mathcal{M}') \tag{8}$$

(ties broken in terms of order in the sequence $\mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_N$). Then the source policies corresponding to $\mathbf{A}$ are $\{\rho_1, \rho_2, \cdots, \rho_c\}$ where ρ_i corresponds to A_i and is the optimal policy of $\mathcal{M}^i$.

The definition is illustrated in Figure 2 and it means the following. $\mathcal{M}^i$ is the element of A_i that minimizes the maximum d_V distance to the other elements of the cluster, and hence, by Lemma 3, is in a worst case sense the best representative of the MDPs in cluster A_i. By choosing the optimal policy of $\mathcal{M}^i$ as a source policy ρ_i, we ensure have the best worst-case representation of the MDPs in each cluster A_i. We make this final statement exact in the next section, in particular in Lemma 5 when we bound the regret of EXP-3-Transfer with respect to optimal policy of *any* MDP in the repository (rather than just the source policies as done in Section 3).

4.2 Cost Function of a Clustering

Let $\mathbf{A} = \{A_1, A_2, \cdots, A_c\}$ be a clustering with the c source policies $\rho_1, \rho_2, \cdots, \rho_c$ chosen as defined in the previous subsection. Our goal in this section will be to quantify the regret of EXP-3-Transfer with respect to any π_k^* when executed on the new task $\mathcal{M}_{N+1}$ for T episodes, where π_k^* is the optimal policy of $\mathcal{M}_k$ in the repository $\mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_N$. We will consider the case where EXP-3-Transfer is run with source policies $\rho_1, \rho_2, \cdots, \rho_c$, and hence this regret will quantify how good the clustering $\mathbf{A}$ is for transfer – in particular, the lower the regret, the more preferable the clustering. As a result, the regret will serve as the cost function for choosing a clustering to derive the source policies. Before proceeding, note that in Theorem 2, we computed the regret between EXP-3-Transfer and ρ_i – in this section we extend those results to derive the regret with respect to π_k^*.

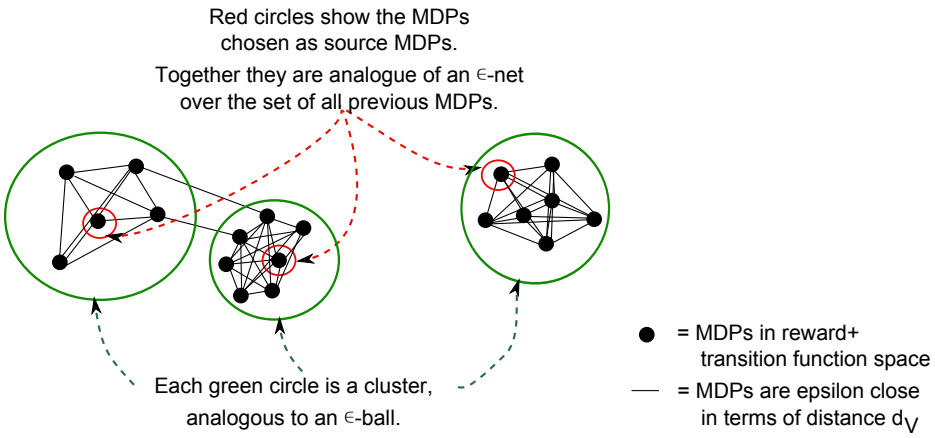


Figure 2: This figure sketches our basic approach to deriving the source policies. The black circles represent our repository of N MDPs. The goal is to put them into c clusters and then derive c source policies from the c source tasks. The figure illustrates the idea for $c = 3$. Each cluster is an analogue of an ϵ-ball in a metric space according to d_V (7). The source MDPs form an analogue of an ϵ-net over the set of previous MDPs with respect to d_V. The function d_V measures how well the policy of one MDP performs in the other – and hence the source policies being an ϵ-net implies that, given any MDP in the repository, there is at least one source policy which has performance that is 'ϵ-close' to the performance of the optimal policy of the previous MDP.

To begin, let the diameter of a cluster A_i, and the mean diameter of the clustering $\mathbf{A}$ be:

$$\epsilon_i \triangleq \max_{\mathcal{M} \in A_i} d_V(\mathcal{M}^i, \mathcal{M}), \quad \bar{\epsilon} \triangleq \frac{1}{N} \sum_{i=1}^{c} |A_i| \epsilon_i \tag{9}$$

So $\bar{\epsilon}$ is the average diameter of the clusters, weighted by the size of the clusters. Therefore, $\bar{\epsilon}$ gives the average distance from a cluster center $\mathcal{M}^i$ to an element of the cluster. As such, $\bar{\epsilon}$ measures how much of the diversity in the repository is captured by the chosen clusters. The smaller $\bar{\epsilon}$, the smaller the average d_V distance between the cluster centers ρ_i and the MDPs in A_i, and hence more of the diversity of policies in the repository is captured by the ρ_i.

Using these, we can give an average case quantification[1]of the performance of a clustering $\mathbf{A}$ when used to generate the source policies and used in EXP-3-Transfer. Let the new MDP $\mathcal{M}_{N+1}$ have transition and reward functions R_{N+1} and P_{N+1}. Define $K_k = \max_{s,a} |R_k(s,a) - R_{N+1}(s,a)|$ and $K'_k = \max_{s,a} |P_k(.|s,a) - P_{N+1}(.|s,a)|$ where R_k and P_k are the reward and transition functions for MDP $\mathcal{M}_k$. Additionally let

$$K(k) \triangleq \frac{K_k + \gamma K'_k R_{max}}{(1-\gamma)^2} \tag{10}$$

We have the following result which derives from Theorem 2.

2. A worst case quantification is also possible. However, the assumptions underlying the worst case seems too weak, and the cost function correspondingly not sufficiently discriminating – i.e. it identifies MDPs that are intuitively dissimilar as being similar. In particular in our experiments, we found this cost function to not give us the intuitive clusters. We discuss this function further in appendix B.

Lemma 5 *If EXP-3-Transfer is run with source policies derived from* $\mathbf{A}$ *using definition 4 with* β *set as in Theorem 2, then for* $\pi_k^* \in A_i$*, such that EXP-3-Transfer did not eliminate* ρ_i *by step* T*, the following is true:*

$$V_{N+1}^{\pi_k^*} - \mathbb{E}_{R,P}\left[\mathbb{E}_{E3T}\left[\sum_{t=1}^{T} \bar{x}_{i_t}(t) \middle| \rho_i \in C_T\right]\right] \leq g(c) + \epsilon_i + 2K(k) \tag{11}$$

Additionally, with probability $1-\delta$*, with respect to randomization due to the target MDP* $\mathcal{M}_{N+1}$*, for each* $\pi_k^* \in A_i$ *such that* ρ_i *was eliminated, there exist* $\rho_{i'}$ *such that*

$$V_{N+1}^{\pi_k^*} \leq V_{N+1}^{\rho_{i'}} + \epsilon_i + 2K(k)$$

The proof is in Appendix A.

We now use the lemma to derive the cost function. First, in the limiting case of $\delta = 1$, none of the policies ρ_i are eliminated and in this case the bound (11) applies to all the MDPs in the repository. In the lemma we assume that each MDP $\mathcal{M}_k$ in the repository is equally likely to be the one with the minimum $K(k)$. Hence, taking the average of (11) over all the MDPs, we get the upper bound of the average regret with respect to all the optimal policies π_k^* of MDPs in the repository:

$$g(c) + \bar{\epsilon} + 2\frac{1}{N}\sum_i K(k) \tag{12}$$

$K(k)$ depends on the repository of $\mathcal{M}_k$s which we do not control or make any assumptions about. Hence, we associate with each cluster $\mathbf{A}$ the parameters $(c, \bar{\epsilon})$, defined at the beginning of this section, and use (12) to define the following cost:

Definition 6 *The cost of a clustering* $\mathbf{A}$ *with parameters* $(c, \bar{\epsilon})$ *is defined to be:*

$$cost(\mathbf{A}) \triangleq g(c) + \bar{\epsilon} \tag{13}$$

4.3 Hardness of finding the Optimal Clustering

In this section we introduce the problem of finding the clustering that minimizes $cost()$. We argue that optimizing the cost function is hard, which sets the stage for developing our discrete optimization algorithm in the next section. Specifically we will show that it is hard to optimize an upper bound $cost_m()$ of $cost()$ where $cost()$ was defined in Definition 6. To that end, define the average max-diameter of a clustering $\mathbf{A}$ to be:

$$\bar{\epsilon}_m = \frac{1}{N}\sum_i \sum_{\mathcal{M}\in A_i} \max_{\mathcal{M}'\in A_i} d_V(\mathcal{M}, \mathcal{M}') \tag{14}$$

Now define,

Definition 7 *Define* $cost_m(\mathbf{A}) \triangleq g(c) + \bar{\epsilon}_m$.

We have the following relationships.

Lemma 8 *The parameter* $\bar{\epsilon}_m$ *of* $\mathbf{A}$ *is an upper bound on the parameter* $\bar{\epsilon}$ *of* $\mathbf{A}$ *defined in (9). Furthermore,* $cost_m(\mathbf{A}) \geq cost(\mathbf{A})$ *for all clusterings* $\mathbf{A}$.

Proof This follows directly from the definition of $\bar{\epsilon}$ – in particular, $\bar{\epsilon}_m$ upper bounds $\bar{\epsilon}$ because for each $\mathcal{M}_j \in A_i$, $\max_{\mathcal{M}' \in A_i} d_V(\mathcal{M}_j, \mathcal{M}') \geq \min_{\mathcal{M}} \max_{\mathcal{M}'} d_V(\mathcal{M}, \mathcal{M}') = d_V(\mathcal{M}^i, \mathcal{M})$. The second part of the lemma now follows by the definitions of $cost()$ and $cost_m()$. ■

We reduce the minimum clique-cover problem (Karp, 1972) to the problem of finding the clustering that minimizes $cost_m$ and hence establish that it is NP-complete. We start by describing the clique cover problem. Let $G = (V, E)$ be a graph where V is the set of vertices and E is the set of edges. A subset $V' \subset V$ is a clique if for any $v, v' \in V$, there is an edge $(v, v') \in E$. The minimum clique cover problem is finding a partition $V_1, V_2, \cdots, V_n$ of V such that each V_i is a clique and n is minimum – that is there exists no other partition with $V'_1, V'_2, \cdots, V'_m$ of V such that each V'_i is a clique and $m < n$. We have the following theorem for $cost_m$.

Theorem 9 *Given a graph $G = (V, E)$, in time polynomial in $|V|$ and $|E|$, we can reduce the minimum clique cover problem for G to finding the clustering $\mathbf{A}^*$ of some set of MDPs $\mathcal{M}_1, \mathcal{M}_2$, $\cdots$, $\mathcal{M}_{|V|}$, with all $\mathcal{M}_i$ defined on the same state and action spaces, where $\mathbf{A}^* \triangleq \arg\min_{\mathbf{A} \in \mathcal{C}} cost_m(\mathbf{A})$.*

The proof is given in Appendix A. Since the clique cover problem is NP-complete, we immediately have the following corollary.

Corollary 10 *Finding the clustering minimizing the upper bound $cost_m()$ of $cost()$ is NP-complete.*

Unfortunately, we do not yet have a proof that minimizing $cost()$ is hard – but the fact that minimizing the upper bound $cost_m()$ is hard, leads us to conjecture that minimizing $cost()$ is hard as well. For this reason, to optimize the $cost()$ function we need a discrete optimization function, which we will develop in the next section.

We end this section by contrasting our approach to previous approaches to clustering MDPs in TLRL (for instance (Wilson et al., 2007)). In prior work, MDPs are typically characterized by a finite number of real valued parameters, and the distance between parameters determine how similar the MDPs are. These MDPs can then be clustered by, for instance, putting the non-parametric Dirichlet process prior over the parameters of all MDPs, and then using Monte Carlo inference methods to find the clustering that maximizes the posterior probability.

Since our notion of similarity between MDPs is based on policies, to apply this approach to our case, we need (1) a relatively compact parametric representation of optimal policies, and (2) a metric that relates the policy-parameters to values of policies and (3) that optimal policies uniquely characterize MDPs. (1) seems difficult for interesting policies, and (2) seems reasonable only for 'linear' domains – i.e. domains where a small change in policy results in a corresponding small change in its value. (3) runs completely counter to one of our main motivations for using policy based clustering, which is that different MDPs might have identical or near-identical optimal policies. So given all these, we were motivated to construct a clustering algorithm adapted to our policy based clustering of MDPS. We also compare our method and the method of Wilson et al. (2007) in Section 7.1, where in a simplified version of their domain, our approach recovers better clusterings.

5. Finding the Optimal Clustering

In Section 4 we defined a cost function that measures how well a particular clustering of the repository of MDPs trades off size and diversity of the set of source policies that are obtained from the

clustering. In Section 4.3 we argued that optimizing $cost()$ is hard. This implies that we need a discrete global optimization algorithm for finding the optimal clustering. So in Section 5.1, we introduce the problem of discrete optimization and our approach to solving it. Then in Sections 5.2 to 5.6 we derive and analyze our general optimization algorithm. Finally, in Section 5.7, we apply the algorithm to the problem of finding the optimal clustering.

5.1 Global Optimization With Stochastic Search

Our goal is to solve the discrete global optimization problem of computing $\arg\min_{A \in \mathcal{C}} cost(A)$, where $\mathcal{C}$ is the set of all possible clusterings of $\mathcal{M}_i$s. Algorithms for discrete optimization problems tend to fall into specific classes, appropriate for the problem at hand. We present a *stochastic search* algorithm for our problem (see Figure 3).

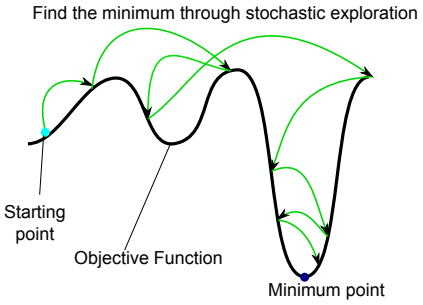


Figure 3: This figure illustrates how stochastic search may be used solve function optimization (minimization in this case). The thick black curve is the *objective function* to be optimized/minimized and the goal is the find the point x at which the curve the attains the minimum value (the blue circle). A stochastic search algorithm starts at a particular point (the cyan circle in this figure) and each time step the it jumps (shown by thin green arrows) to a new candidate point, chosen according to some stochastic strategy. The arrows shown are one possible run of the stochastic search algorithm, with different runs likely going through different sets of points. The candidate may move towards and away from the minimum point. This kind of optimization is necessary when the objective function does not have nice properties (like convexity) and standard algorithms (like gradient descent) are not applicable.

Our basic strategy is to construct a distribution over $\mathcal{C}$, that concentrates around the optimum and around clusterings with low cost. The concentration property implies that if we repeatedly sample from this distribution, we will find the optimum or a good/low cost clustering with high probability. However, in general, exact sampling from such distributions is difficult, and so our algorithm samples approximately from this distribution using a Markov chain Monte Carlo approach – see (Robert and Casella, 2005) for a comprehensive introduction to MCMC and Metropolis Hastings Markov chains (MH chain in short) that we use. The use of MCMC turns our algorithm into a stochastic search algorithm.

The resulting algorithm can be understood as simulated annealing (Kirkpatrick et al., 1983) but with *stochastic temperature changes*. Figure 4 describes simulated annealing qualitatively and also contrasts it with our algorithm, MHAV. Simulated annealing is a stochastic search algorithm for optimizing an objective function f, where the search rule at step t jumps from point y to y' with

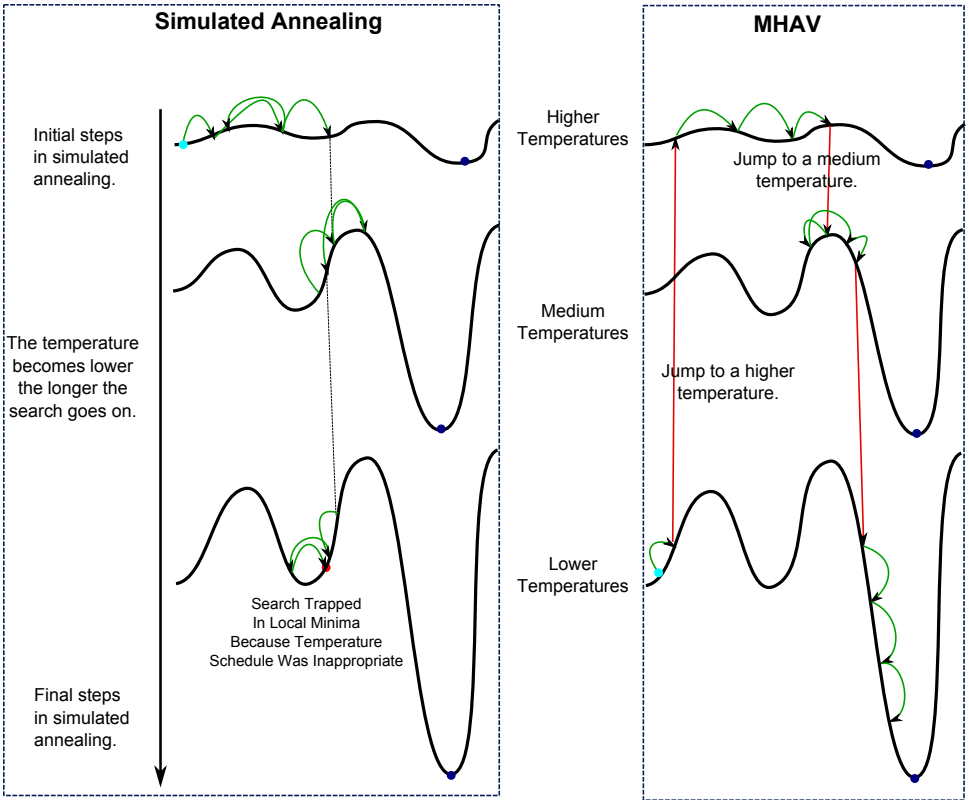


Figure 4: This figure compares simulated annealing and our MHAV algorithm for finding minimum of an objective function f. The green arrows show the search steps taken by the algorithms. Simulated annealing decreases a *temperature* parameter over time, which effectively results in the function surface becoming effectively less and less flat. The figure illustrates a search that has failed due to incorrect temperature decrease schedule. Unlike simulated annealing, MHAV jumps between temperature values stochastically. The green arrows are search steps for a fixed temperature, while the red arrows are jumps between temperatures. See beginning of Section 5.1 for a full description.

probability $P_t(y, y') = q_{y,y'} \exp[-\frac{1}{z(t)} \max(0, f(y') - f(y)]$. $q_{y,y'}$ is a candidate distribution (probability with which y' is proposed as the next point given current point x) and the remaining term depends on the improvement $f(y') - f(y)$ and the *temperature* $z(t)$ (see below). Because of the $\max$ in the improvement term, jumps to lower f-valued (i.e. better) points y' succeed with probability 1, while jumps to higher f-valued (i.e. worse) points y' succeed with probability depending on how much higher (worse) the point is.

The temperature $z(t)$ is a chief feature of simulated annealing, it is user defined and determines practical success of the algorithm. It is a sequence decreasing in t and in essence changes how easy it is to explore the objective function surface – with higher temperature allowing the search to travel over longer distances and in effect making the objective function flatter, and lower temperature restricting the search to points local to the current point, in effect making the objective function steeper (Figure 4 illustrates three qualitatively different temperatures). The temperature sequence $z(t)$ needs to be set very carefully so that during the higher temperature phase the search travels

longer distances to the places where the minimum point is, and then during the lower temperature phases, the search moves in a 'local neighborhood' of the minimum point and and finds the point itself or a point close to it. Clearly, this is a fairly difficult problem and requires some understanding of the objective function.

Figure 4 illustrates our algorithm, MHAV. The algorithm is similar to simulated annealing in that the search rule is such that jumps to better points succeed with probability 1, while jumps to worse points succeed with probability depending on how much worse the point is. It also uses temperatures to modify the flatness of the function (in the algorithm description in Section 5.2 the temperature parameter is represented by λ, with high temperature corresponding to low λ). However, unlike simulated annealing, MHAV does not use a fixed temperature schedule but moves between different temperatures stochastically (Figure 4 illustrates the algorithm jumping between three qualitatively different temperatures). In essence MHAV searches through the augmented space (λ, y) and the search rule has the form $P_t[(\lambda, y), (\lambda', y')]] \approx q_{(\lambda,y),(\lambda',y')} \exp\{\max[0, -(f(y') \ln \lambda' - f(y) \ln \lambda]\}$. That is, MHAV searches over both the temperature and the solution space simultaneously and so avoids the very difficult problem of needing to set the temperature schedule. Convergence is still guaranteed by convergence of the Metropolis-Hastings algorithm (see Section 5.3). Our proof of the convergence, and the speed of convergence of the algorithm is simple compared to simulated annealing (see, for instance, (Locatelli, 2000) for contrast).

We present our algorithm in steps. We first cast the discrete optimization problem as the problem of sampling from a specific distribution derived from the objective function (Section 5.2), then we present the general Metropolis-Hastings scheme for approximate sampling from a distribution (Section 5.3), and after that we present and analyze our adaptation of Metropolis-Hastings for discrete optimization (Sections 5.4, 5.5 and 5.6), which we call MHAV (Metropolis-Hastings with a auxiliary variable). The MHAV algorithm is a general optimization algorithm, which we then adapt to the problem of searching for the optimal clustering (Section 5.7).

5.2 Optimization as Sampling

In this section, we show how to convert the problem of global optimization to the problem of sampling from a distribution. The method we discuss was inspired by the basic idea behind simulated annealing (Kirkpatrick et al., 1983). Assume that our goal is to minimize a cost function f defined over some finite set Y. In particular assume that there is a subset $\hat{Y} \subset Y$ for which $y \in \hat{Y}$ has acceptable cost $f(y)$. Let $\Lambda = \{\lambda_1, \lambda_2, \cdots, \lambda_n\}$, $\lambda_i < \lambda_{i+1}$ such that $\exists \hat{\Lambda} \subset \Lambda$ which satisfies

$$\sum_{\lambda \in \hat{\Lambda}, y \in \hat{Y}} \lambda^{-f(y)} \geq \theta > 0 \tag{15}$$

We now define the distribution

$$\bar{\Pi}(\lambda, y) \triangleq \lambda^{-f(y)} Z^{-1} \tag{16}$$

where $Z \triangleq \sum_{y,\lambda} \lambda^{-f(y)}$ is the normalization term. Given the existence of $\hat{\Lambda}$, if we draw repeatedly from $\bar{\Pi}$, then after t draws, with probability at least

$$1 - (1 - \theta)^t \tag{17}$$

we will draw an element (λ, y) where $y \in \hat{Y}$. Since $1 - \theta < 1$, the probability that we draw an element from $y \in \hat{Y}$ goes to 1 at rate $(1 - \theta)^t$ (hence, the closer θ is to 1, the faster the convergence

rate will be). So this solves the solves the discrete global optimization problem of optimizing the function f.

The reason we introduce the parameter λ rather than just sampling from $Pr(y) \triangleq \frac{f(y)}{\sum_{y'} f(y')}$ is because this distribution is typically intractable to sample from. Indeed, for our MDP clustering problem, the function f is $cost()$ and the set Y is the set of clusterings $\mathcal{C}$. By the the hardness result for $cost()$ in Appendix A, we conjecture that sampling from $Pr(y)$ is in fact intractable. Hence, we need to use approximate sampling methods.

Use of approximate sampling turns sampling from $Pr(y)$ into a stochastic search over the objective function $f()$ to find the global minima. As detailed in the previous subsection, one powerful way to augment such search methods is to modify the objective function by introducing temperatures to change the flatness of the function, and the λ parameter serves precisely this purpose. In particular, when λ is high (corresponding to 'low temperature'), the modified objective function $\lambda^{-f(y)}$ is 'steep', and when λ is low (corresponding to 'high temperature') $\lambda^{-f(y)}$ is 'flat'.

Given this motivation, in Section 5.3, we present the Metropolis-Hastings (MH) algorithm that may be used to approximately sample from arbitrary distributions, and then in Section 5.4 we adapt the MH algorithm to sample from $\bar{\Pi}$.

5.3 Sampling Using Metropolis-Hastings Chains

In this subsection we describe a standard method to approximately sample from a distribution Π over a large finite space $\mathcal{X}$. In the next section we will use this method to sample from $\bar{\Pi}$ and complete our global optimization method. We use upper-case Roman letters for random variables and lower-case letters to refer to their realized values. In the following, we use the theory of Markov chains as found in (for instance) (Levin et al., 2009). A *Markov chain* over a (finite) state-space $\mathcal{X}$ is stochastic process X_n taking values in $\mathcal{X}$ such that $Pr(X_n = x|x_0, x_1, x_2, \cdots, x_{n-1}) = P_n(x|x_{n-1})$. The distribution $P_n(\cdot|\cdot)$ is called the *transition kernel* for the chain, and can be represented by a $|\mathcal{X}| \times |\mathcal{X}|$ matrix, also denoted by P_n, such that the entry (x, y) is $P_n(y|x)$ (here we have identified each element of $\mathcal{X}$ with an integer in $\{1, 2, \cdots, |\mathcal{X}|\}$ in some order). A Markov chain is said to be time-homogeneous if $P_n(x'|x) = P(x'|x)$, i.e. P_n is independent of time n. We will only consider time-homogeneous chains. A distribution Π over $\mathcal{X}$ is said to be *stationary* for the chain with kernel P if it satisfies:

$$\Pi(x') = \sum_x \Pi(x) P(x'|x) \tag{18}$$

That is if P is stationary for Π, then, if we draw x according to $\Pi(x)$, then choose x' according to $P(x'|x)$, then the distribution over x' will also be $\Pi(x')$.

Let $P^n(x'|x)$ be the probability that $X_n = x'$ given that $X_0 = x$, that is

$$P^n(x'|x) = \sum_{x_1, x_2, \cdots x_{n-1}} \prod_{i=0}^{n-1} P(x_{i+1}|x_i), \qquad \text{where } x_0 = x, x_n = x'$$

Then the chain X_n (equivalently, the kernel $P(\cdot|\cdot)$) is said to be *irreducible* if for each $x, x' \in \mathcal{X}$, $\exists n$ with $P^n(x'|x) > 0$. It is called *aperiodic* if the set $\{n : P^n(x|x) > 0\}$ has greatest common divisor 1 – that is there is no period to the set of time steps at which the chain returns to some state x, starting from x itself.

Theorem 11 *The following are true for any aperiodic and irreducible Markov chain with kernel P:*

1. *P has a stationary distribution Π and for any $y \in \mathcal{X}$,*

$$\lim_{n\to\infty} \|P^n(\cdot|y) - \Pi(\cdot)\|_{\mathrm{TV}} = 0 \tag{19}$$

   where $\|P - P'\|_{\mathrm{TV}} = \sup_{A\subset\mathcal{X}} |P(A) - P'(A)|$ is the total variation distance between any two distributions P, P' over $\mathcal{X}$.

2. *If Π is stationary for P and $\|P^n(\cdot|y) - \Pi(\cdot)\|_{\mathrm{TV}} \leq k$ then $\|P^{n'}(\cdot|y) - \Pi(\cdot)\|_{\mathrm{TV}} \leq k$ for all $n' > n$.*

Proof For the first part and second part, see (for instance), respectively, Theorem 4.9 and Lemma 4.12 in (Levin et al., 2009)). ■

This result is important because it can be used to approximately sample from a distribution Π that is hard to sample from directly. The idea is to construct a Markov chain X_n with stationary distribution Π. Theorem 11 implies that if we simulate X_n long enough, then eventually we will start sampling from Π. To that end, the Metropolis-Hastings chain (MH chain in short) gives a standard way to define such a chain given Π as input (see Robert and Casella (2005) for an in-depth introduction).

A MH chain is defined via an irreducible kernel $\phi(x'|x)$ over $\mathcal{X}$ and an acceptance probability $\mathsf{Acc}_x(x')$. ϕ is problem dependent while Acc is defined as follows:

$$\mathsf{Acc}_x(x') \triangleq \min\left\{1, \frac{\phi(x|x')\Pi(x')}{\phi(x'|x)\Pi(x)}\right\} \tag{20}$$

Given this, the MH chain has transition

$$P_{MH}(x'|x) = \begin{cases} \phi(x'|x)\mathsf{Acc}_x(x'), & \text{if } x \neq x' \\ 1 - \sum_{x'\neq x} \phi(x'|x)\mathsf{Acc}_x(x'), & \text{otherwise}, \end{cases} \tag{21}$$

It can be easily checked that P_{MH} satisfies the *detailed balance* equation $\Pi(x)P_{MH}(x'|x) = \Pi(x')P_{MH}(x|x')$ which in turn is equivalent to (18) (which can be seen by summing both sides over x'). So P_{MH} is a chain which if simulated long enough will sample from the *target distribution* Π. We will now derive a version of this chain to sample approximately from $\bar{\Pi}$

5.4 Optimization using Metropolis Hastings With Auxiliary Variables (MHAV)

In this section, we show how we may use the MH algorithm to sample from the distribution $\bar{\Pi}$ defined in Section 5.2 and hence perform optimization. We call this algorithm MH with auxiliary variables (MHAV) because we introduced the auxiliary variable λ to enable us to perform global optimization. As we discussed above, MHAV may be thought of as simulated annealing without a temperature schedule.

To adapt MH for our global optimization problem, we set $\mathcal{X} \triangleq Y \times \Lambda$ and set our target to be $\Pi = \bar{\Pi}$. We briefly note here that if we plug $\bar{\Pi}$ into (20), then the normalization term cancels out, and in our algorithms there will never be any need to compute Z. Let ϕ_Y be any irreducible kernel over Y (this will depend on the nature of Y and will be an input to the optimization algorithm – we

discuss this below). Define the following transition kernel over $\Lambda = \{\lambda_1, \lambda_2, \cdots, \lambda_n\}$, parametrized by $\alpha' \in (0,1)$:

$$\phi_\Lambda(\lambda'|\lambda) \triangleq \begin{cases} \alpha' & \text{if } \lambda = \lambda_i, \lambda' = \lambda_{i+1} \text{ and } i < n \\ 1-\alpha' & \text{if } \lambda = \lambda_i, \lambda' = \lambda_{i-1} \text{ and } i > 1 \\ 1 & \text{if } \lambda = \lambda_0, \lambda' = \lambda_1 \text{ or } \lambda = \lambda_n, \lambda' = \lambda_{n-1} \\ 0 & \text{otherwise} \end{cases}$$

Since $\alpha' > 0$, it is easy to see that for any λ_i, λ_j, there is a sequence $\lambda_i, \lambda_{i_1}, \cdots, \lambda_{i_n}\lambda_j$ with positive probability under ϕ_Λ. That is,

Lemma 12 *ϕ_Λ is irreducible.*

Given the above, the proposal distribution $\bar{\phi}[(\lambda', y')|(\lambda, y)]$ for $\bar{P}_{\mathrm{MH}}$ is defined using the parameters $\alpha, \beta \in (0,1)$, $\alpha + \beta < 1$, as follows.

$$\bar{\phi}[\lambda', y'|\lambda, y] \triangleq \begin{cases} \alpha\phi_\Lambda(\lambda'|\lambda) & \text{if } \lambda \neq \lambda', y = y' \\ \beta\phi_Y(y'|y) & \text{if } \lambda = \lambda', y \neq y' \\ (1-\alpha-\beta) + \beta\phi_Y(y'|y) & \text{otherwise} \end{cases} \tag{22}$$

The transition kernel $\bar{P}_{\mathrm{MH}}$ is now defined as in (21) using $\bar{\phi}$ as the proposal distribution and (16) as the target distribution:

$$\bar{P}_{\mathrm{MH}}(x'|x) = \begin{cases} \bar{\phi}(x'|x)\bar{\mathsf{Acc}}_x(x'), & \text{if } x \neq x' \\ 1 - \sum_{x' \neq x} \bar{\phi}(x'|x)\bar{\mathsf{Acc}}_x(x'), & \text{otherwise}, \end{cases} \tag{23}$$

Given the above, the overall discrete global optimization algorithm MHAV (Metropolis-Hastings with Auxiliary Variable) is listed in Algorithm 2. Note that lines 5-6 are sampling from the transition kernel $\bar{P}_{\mathrm{MH}}$.

Algorithm 2 MHAV$(\Lambda, Y, \bar{\Pi}, \bar{\phi}, T_M)$

1: **Input:** The set of auxiliary variables Λ, the search space Y, the target distribution $\bar{\Pi}$, and proposal distribution $\bar{\phi}$, T_M number of iterations to run algorithm.
2: **Output:** An element $y \in Y$.
3: **Initialize:** Initial, $\lambda(0) = \lambda_0$, $y(0) =$ arbitrary element of Y.
4: **for** $t = 1$ to T_M **do**
5: Sample $(\lambda', y') \sim \bar{\phi}[\cdot|\lambda(t), y(t)]$
6: With probability $\bar{\mathsf{Acc}}_{\lambda(t),y(t)}[\lambda', y']$, set $\lambda(t+1) = \lambda'$, $y(t+1) = y'$, and with probability $1 - \bar{\mathsf{Acc}}_{\lambda(t),y(t)}[\lambda', y']$, set $\lambda(t+1) = \lambda(t)$, $y(t+1) = y(t)$.
7: **end for**
8: **return** $\arg\min_{y(t)} f(y(t))$.

5.5 Analysis of the MHAV Algorithm

The reader may prefer to skip this and the next section and move directly to Section 5.7, which details how the MHAV algorithm is adapted to search for the optimal cluster. We begin analysis of our algorithm by showing that the kernel $\bar{P}_{\mathrm{MH}}$ for the $\bar{\phi}$ defined above is indeed irreducible and aperiodic.

Lemma 13 *If $\bar{\Pi}$ and ϕ_Y satisfy $\min_{x,x'} \frac{\bar{\Pi}(x')\phi_Y(x|x')}{\bar{\Pi}(x)\phi_Y(x'|x)} > b > 0$, the kernel $\bar{P}_{\mathrm{MH}}$ is irreducible and aperiodic.*

The proof is given in Appendix A – additionally note that by finiteness of f and λ_is, and irreducibility of ϕ_Y, such a b always exists. The following theorem establishes the probability with which we draw an element from the acceptable set $\hat{Y}$ when using $\bar{P}_{\mathrm{MH}}$ to sample.

Theorem 14 *$\bar{P}_{\mathrm{MH}}$ has $\bar{\Pi}$ as its stationary distribution, and hence for any initial state x_0 of the chain $\bar{P}_{\mathrm{MH}}$,*

$$\lim_{n\to\infty} \|\bar{P}^n_{\mathrm{MH}}(\cdot|x_0) - \bar{\Pi}(\cdot)\|_{\mathrm{TV}} = 0 \tag{24}$$

In particular if at step t $\|\bar{P}^t_{\mathrm{MH}}(\cdot|x_0) - \Pi(\cdot)\|_{\mathrm{TV}} \leq k$, then $\bar{P}^{t'}_{\mathrm{MH}}(x \in \hat{\lambda} \times \hat{Y}|x_0) \geq \theta - k$ for all $t' > t$.

The proof is given in Appendix A. Combining the above with (17) (and the discussion following the equation) shows that the probability that MHAV samples from the acceptable set goes to 1 at rate $> 1 - \theta + k$ from step t_k onwards, where t_k is the step such that $t > t_k$ implies $\|\bar{P}^t_{\mathrm{MH}}(\cdot|x_0) - \Pi(\cdot)\|_{\mathrm{TV}} \leq k$.

We can also derive a convergence rate which establishes that for every k such a t_k exists and the rate of convergence of MHAV goes arbitrarily close to $1 - \theta$. Define the *diameter* of $\mathcal{X}$ given the Markov chain $\bar{P}_{\mathrm{MH}}$ to be

$$D \triangleq \min\{l|\forall x, x', \bar{P}^l_{\mathrm{MH}}(x'|x) > 0\} \tag{25}$$

Now define the ratio

$$\delta \triangleq \min_{x,x'} \frac{\bar{P}^D_{\mathrm{MH}}(x'|x)}{\bar{\Pi}(x')} \tag{26}$$

D is finite and δ non-zero by the irreducibility of $\bar{P}_{\mathrm{MH}}$ and finiteness of f. We have the following:

Theorem 15 *The convergence rate of $\bar{P}_{\mathrm{MH}}$ to $\bar{\Pi}$ satisfies:*

$$\|\bar{P}^n_{\mathrm{MH}}(\cdot|x_0) - \bar{\Pi}(\cdot)\|_{\mathrm{TV}} \leq (1-\delta)^{n/D}$$

for any initial state x_0.

This derives directly from the proof of Theorem 4.9 (Levin et al., 2009) and is given in Appendix A. This implies that for each k, we have $t_k = D \ln k/(1-\delta)$.

5.6 Setting the Optimization Parameters

We now discuss heuristics to set the parameters α' in ϕ_Λ and α, β in $\bar{P}_{\mathrm{MH}}$ so as to optimize the convergence rate derived above, by minimizing δ (defined in (26)). In setting these parameters, we are given the proposal distribution ϕ_Y, which was required to be an irreducible kernel on Y, and the target distribution $\bar{\Pi}$ over $\Lambda \times Y$. We start with the following result which simplifies deriving our result

Lemma 16 *D (defined in (25)) is independent of α', α, β.*

The proof is given in Appendix A.

Corollary 17 *Given f, ϕ_Y, the set of paths of positive probability under $\bar{P}_{\mathrm{MH}}$ is invariant with respect to α', α, β.*

Proof Follows directly from the proof of Lemma 16. ■

So, we need to set α', α, β to maximize δ. However, δ also depends on f and ϕ_Y, both of which are unknown and so it is difficult to specify optimal values for these a-priori. However, we can give heuristic arguments for setting these parameters in terms of increasing the 'flow' of the search process through the search space Y. First, α' is used to choose whether we should increase or decrease the λ value. We set α' to $1/2$ to ensure a neutral value and that we do not favor either direction and ensure maximum flow through the search space.

Now note that at each step the chain $\bar{P}_{\mathrm{MH}}$ moves either in Λ space or Y space. α and β determine, respectively, how often we move in the Λ and how often in Y. To make the search more effective (based on analysis of simulated annealing type algorithms), it seems we need to make sure that initially we need to explore the Y quite a bit and only settle down after we have explored sufficiently, by increasing the λ_i value. Hence, our recommendation is to set the α to be significantly smaller than β, ideally the ratio α/β should reflect how difficult we expect it to be to get close to the best y^* (with smaller ratio for greater difficulty). Even though our parameter settings are heuristic, we again stress that this only affects the convergence speed, but not the ultimate convergence. This is in contrast to simulated annealing where convergence itself is guaranteed only if we set the parameters carefully.

5.7 Searching for the Optimal Cluster

Searching for the optimal cluster can now be be solved using the MHAV algorithm. The algorithm for searching through the space of clusterings is given in Algorithm 3. In this case, $Y = \mathcal{C}$, and the objective function is $f(\mathbf{A}) = cost(\mathbf{A})$. To complete the specification of MHAV for our problem, we define the distribution $\phi_Y^M(\mathbf{A}'|\mathbf{A})$ to be the probability with which a randomized procedure converts $\mathbf{A}$ to $\mathbf{A}'$. The randomized procedure is as follows.

Given $\mathbf{A} = \{A_1, \cdots, A_n\}$, choose A_i uniformly at random, and A_j uniformly at random from $(\mathbf{A} - \{A_i\}) \cup \{B\}$, where B is place-holder/empty set representing a new cluster. Now choose k_i points from A_i, uniformly at random (without replacement), and put them in A_j. The clustering resulting from this transfer is $\mathbf{A}'$.

Note that if $A_j = B$, then $\mathbf{A}'$ has one more cluster than $\mathbf{A}$. Additionally, if $k_i = |A_i|$, and $A_j \neq B$, then $\mathbf{A}'$ has one less cluster than $\mathbf{A}$. Otherwise, they both have the same number of

clusters, but differing at A_i and A_j. The number k_i is chosen using the exponential distribution over $\{1, 2, \cdots, |A_i|\}$:

$$PE(k; \theta_1) \triangleq e^{-k\theta_1}(e^{\theta_1} - 1)/(1 - e^{|A_i|\theta_1}), \text{ where } \theta_1 > 0$$

This ensures that k_i is small with higher probability and so we are less likely that $\mathbf{A}$ and $\mathbf{A}'$ are very different due to moving large number of points from A_i to A_j. The parameter θ_1 is user dependent and in our experiments we set it to 1.

The following Lemma shows that $\phi_Y^M(\mathbf{A}'|\mathbf{A})$ is irreducible and hence satisfies the condition in Lemma 13 and hence ensures the convergence results in Section 5.5.

Lemma 18 *ϕ_Y^M defined above is irreducible.*

The proof is in Appendix A.

Algorithm 3 Search-Clusterings($\mathsf{M}, d, \Lambda, T_M$)

1: **Input:** A set of MDPs $\mathsf{M} = \{\mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_N\}$, the set of auxiliary variables Λ, a cost function $cost$, input condition $term$.
2: **Initialize:** ϕ_Y^M defined with respect to M; $\bar{\phi}$ defined using ϕ_Y^M using (22); define $\bar{\Pi}(\lambda, \mathcal{M}) = \lambda^{-cost(\mathcal{M})}$.
3: **return** MHAV($\Lambda, \mathsf{M}, \bar{\Pi}, \bar{\phi}, T_M$)

Algorithm 4 Continual-Transfer($d, \Lambda, cost, T_M, l, \Delta R, \beta, T$)

1: **Input:** A metric d, which is either a d_M or d_V; cost function $cost$, which is either $cost_1$ or $cost_2$; Search-Clustering parameters $T_M, l, \Delta R$, EXP-3-Transfer parameters β, T.
2: **Initialize:** Initial clustering $\mathbf{A} = \emptyset$, collection of previous MDPs M.
3: **for** $h = 1$ to ∞ **do**
4: Get unknown MDP $\mathcal{M}_h$ from the environment and run EXP-3-Transfer($\mathcal{M}_h, sourcePol(\mathbf{A}, d, cost), \beta, T_M, l$).
5: Set $\mathsf{M} \leftarrow \mathsf{M} \cup \{\mathcal{M}_h\}$
6: **if** $h \mod J = 0$ **then** $\mathbf{A}$ = Search-Clusterings($\mathsf{M}, \Lambda, cost, T_M$).
7: **end for**

6. The Continual Transfer Algorithm

In this brief section we combine all the algorithms presented so far into the full continual transfer algorithm, which is listed as Algorithm 4. The algorithm runs in phases and in each phase it solves a MDP using the EXP-3-Transfer algorithm and the current set of source policies as input. In line 4, the function $sourcePol(\mathbf{A}, d, cost)$ generates the c source policies $\rho_1, \rho_2, \cdots, \rho_c$ from clustering $\mathbf{A}$ such that ρ_j is the optimal policy for $\mathcal{M}^j$ where $\mathcal{M}^j$ is chosen from A_j according to (8). If the current phase h satisfies $h \mod J = 0$, then it runs the Search-Clustering algorithm to find a new set of source tasks from the h tasks solved so far.

7. Experiments

We performed three sets of experiments to illustrate various aspects and the efficacy of our algorithm[2]. The baseline algorithm for comparison in the first domain was a multi-task hierarchical Bayesian reinforcement learning algorithm proposed by Wilson et al. (2007), and hence compares our approach to an alternative approach to clustering tasks. In the latter two domains we compare against Probabilistic Policy Reuse (PPR) as introduced by Fernandez et al. (2006) which, as mentioned in Section 1.1, is the main prior work on policy reuse algorithms.

In the larger second and third experiments, we report results for our proposed EXP-3-Transfer algorithm, PPR, and standard Q-learning. Additionally, we experiment using different forms of clustering, including no clustering, our Search-Clustering algorithm, clusters chosen manually by hand, and a greedy clustering procedure. The greedy clustering algorithm is to choose a threshold for the d_V distance, then construct clusters by selecting an MDP arbitrarily to seed a cluster, and finally adding all the MDPs with distance less than that threshold to that cluster. In our graphs, we present the results for the best/lowest cost clustering found by using various threshold values.

A summary of the algorithm combinations used are shown in Table 1.

Algorithm	FULL	SANS	HAND-PICKED	GREEDY
EXP-3-TRANSFER	✓	✓	✓	✓
PROBABILISTIC POLICY REUSE	✓	✓	✓	✓
Q-LEARNING	N/A	N/A	N/A	N/A

Table 1: EXPERIMENT SETUP MATRIX. This table shows the combinations of the algorithms and clustering methods used in the experiments. 'Full' refers to the use of Search-Clustering, 'sans' means without any kind of clustering, 'hand-picked' means using a set of source policies that we selected believing to be optimal, and 'greedy' refers to the heuristic threshold-based method (see text for details).

For each graph presented in experiments 2 and 3, the results are averaged over 10 different target tasks with 10 trials per task. The various parameters used for the clustering and transfer algorithms are given in Table 2.

We present results from experiments run on three different domains. In Section 7.1 we use the coloured grid world domain, commonly used in Bayesian reinforcement learning, to compare the clusters found by our proposed algorithm with those of a hierarchical Bayesian method, in order to motivate our clustering approach. In Section 7.2 we present results from a simple windy corridor domain to demonstrate the clusters produced by our Search-Clustering algorithm. The results show that the clusters found are intuitive in nature. In Section 7.3.2, we present results on the more complex surveillance domain (described briefly in the Section 1) which is a variant of the kinds of problems that are considered, for instance, by An et al. (2012). In this experiment we show the performance for the algorithm combinations given in Table 1 under various numbers of previous tasks and task lengths.

2. The code used in both of the experiments, as well as all of the generated data, can be found here: `http://wcms.inf.ed.ac.uk/ipab/autonomy/code/MDP_Clustering_code.zip`

EXP-3-TRANSFER		SEARCH-CLUSTERINGS					RL
T	δ	α	β	α'	θ	T_M	γ
$1000, 5000$ and 10^4	0.1	0.1	0.8	0.5	1	10^5 (for each of 20 random restarts)	0.9

Table 2: VALUES OF ALGORITHM PARAMETERS. This table gives the values for the different parameters used in our various algorithms. RL refers to common parameters for reinforcement learning algorithms. The T parameter was chosen to illustrate effect of this parameter on algorithm performance. The δ parameter was chosen to allow for a high degree of confidence in the performance of EXP-3-Transfer. The α, α' and β parameters where chosen according to Section 5.6, and θ was chosen heuristically. The value of T_M was selected because we found that restarting gave better results. γ was chosen as appropriate for these domains.

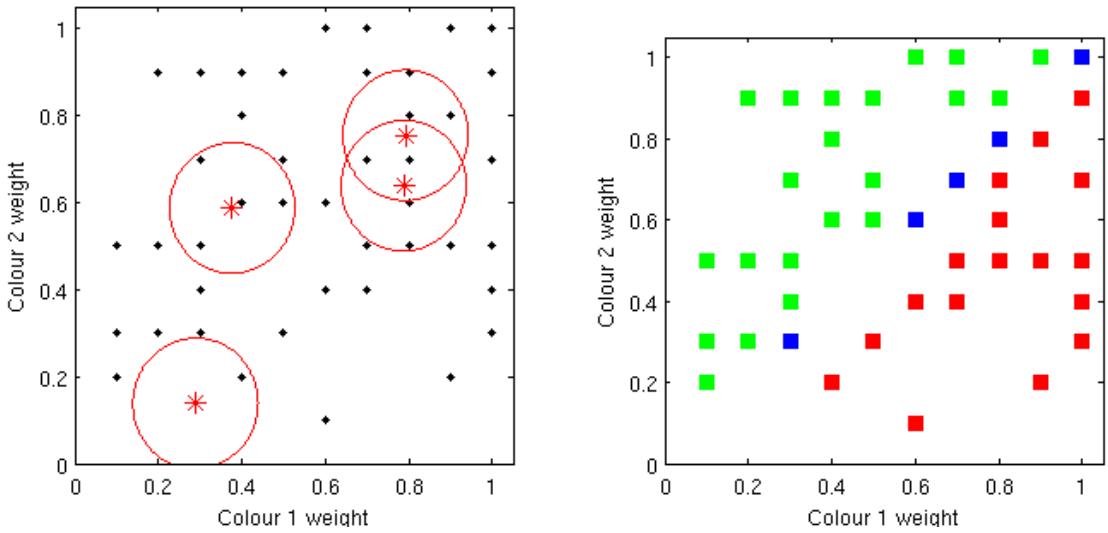
7.1 Coloured Grid World Domain

Wilson et al. (2007) introduced a method for modelling the unknown distribution of tasks in a multi-task reinforcement learning problem using a hierarchical infinite mixture model. Their two-layer model is capable of representing a previously unknown number of classes of MDPs, as well as finding the latent distribution of parameters in each class. For any new task, the model acts as a prior over the parameter space, which enables the generation of a faster initial solution, to be subsequently optimised with additional task-specific learning.

This model describes latent structure in the parameter space of the observed tasks. Instead, our proposed clustering algorithm seeks to identify similarities in policies across different tasks.

To compare this infinite mixture model approach with ours, we consider a simplified version of the coloured maze domain of Wilson et al. (2007), where each cell of a grid world is coloured with one of two possible colours. Each colour is assigned a weight $w \in [0, 1]$, the values of which differ between tasks. The agent navigates the grid world from one corner to the diagonally opposite corner. This is done using four actions, each of which moves the agent to an adjacent cell in the corresponding cardinal direction, deterministically. After every movement, the agent receives a reward equal to the sum of the colour weights of the current cell and the adjacent four cells. The goal of the agent is to maximise the received rewards. Each task is completely described by two parameters (the colour weights) in the unit square. This is intended to simplify visualisation of the resulting clusterings.

We sample 50 tasks uniformly at random from the unit square and cluster them using both the hierarchical infinite mixture model and our framework. Figure 5a shows the clustering results of Wilson et al. (2007), and Figure 5b shows the clusters obtained by our method. Note that all the tasks that lie on the same line that goes through the origin are equivalent as they have the same colour weight ratio, and thus have the same optimal policy with scaled values. As a result, our method is able to model this fact and summarise the complete set of MDPs into a reduced set of landmark MDPs with equivalent policies. On the other hand, modelling the similarities between tasks based on task parameters will fail to realise this equivalence, and subsequently result in many clusters with similar policies because they share a local neighbourhood in parameter space.



(a) Clustering of Wilson et al. (2007)

(b) Clustering of our proposed framework

Figure 5: The tasks in parameter space (the weights of the two colours) of the coloured maze domain. Each dot corresponds to a randomly sampled task. (a) Clusters obtained using a hierarchical infinite mixture model. The asterisks are the means of recovered clusters and the circles show three standard deviations. (b) Clusters obtained using our clustering algorithm. Each colour represents a different class. Note how our method captures the similarity in policy much more closely than methods that cluster directly in parameter space.

7.2 Windy Corridor Domain

The windy corridor domain is illustrated in Figure 6. The domain consists of a row of 10 parallel corridors with a 'wind' blowing from the South to the North along two columns of cells near the entrance to the corridors.

The agent has one action for each possible cardinal direction which moves it in that direction deterministically. In a windy cell, the motion of the agent becomes probabilistic with the probability of moving North being p, and moving in the desired direction being $1 - p$. p depends on the probability of wind in that cell, which is a task parameter ranging from 0 to 0.9.

The MDPs in the domain are distinguished by two values: the location of the goal state and the probability of wind. There are 10 possible wind probabilities, which together with the 10 possible goal locations, results in a total of 100 possible MDPs.

For this domain, we learned the optimal value function for each of the MDPs, and from that computed the distance between every pair of MDPs. This was then used to cluster the MDPs using the Search-Clusterings algorithm. Figure 7 presents the final clusters we found for this domain. This figure shows that the best clustering found by the algorithm placed tasks with the same goal state in the same cluster. This follows intuition because, despite the wind probability, the policies required for MDPs with identical goal states will be identical, yet be different for tasks with different goal states. This demonstrates that the Search-Clusterings algorithm is capable of recovering the expected clusters, thereby providing a sanity check for our algorithm, although we note that the results are stochastic and vary between runs.

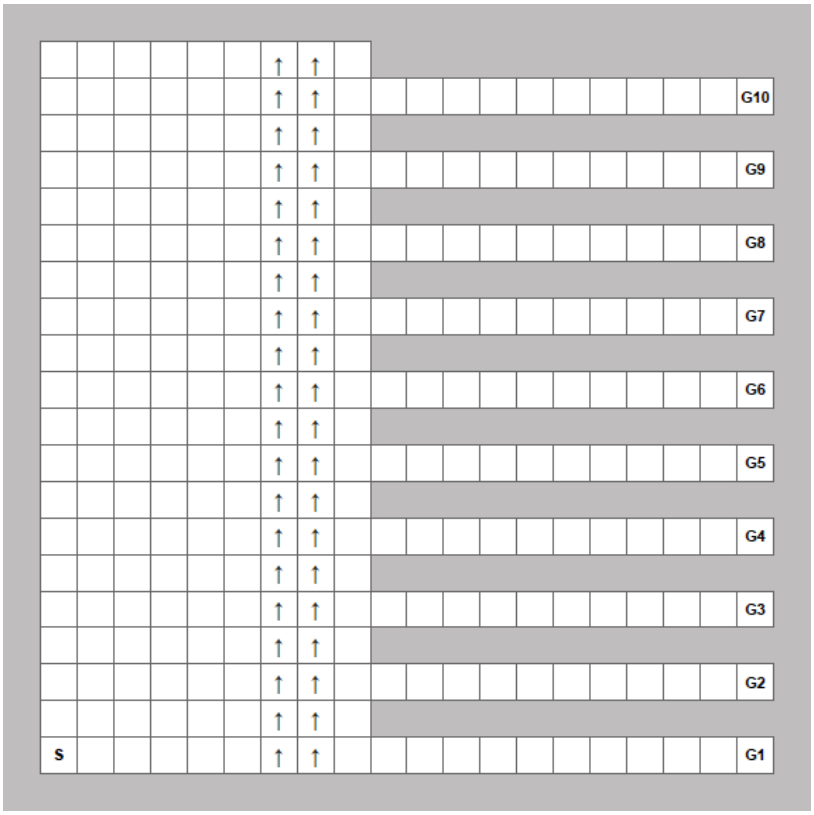


Figure 6: THE WINDY CORRIDOR DOMAIN. This shows the 10 corridors, the location of the goal states (G1-G10) and the direction of the wind (the small arrows). The start state is marked by **s**.

In order to illustrate the effects of the clustering, we incrementally built up the full set of MDPs, by presenting them in a random order to the Search-Clustering algorithm, and having it cluster them after the addition of each new MDP. These results are shown in Figure 8. Note that the allocation of MDPs to clusters remains largely consistent across the presentation of 100 MDPs. In this case, the algorithm recovers 12 main clusters.

7.3 Surveillance Domain

The surveillance domain is illustrated in Figure 9. In this domain, the goal of the agent is to catch infiltrators who wish to break into a target region. There are L different vulnerable locations (abbreviated *v-locations*) in the domain, and the infiltrators only choose a subset of those v-locations to infiltrate – we call these *target v-locations*. The type of the infiltrators is defined by the sequence in which they visit the target v-locations and the goal of the agent is to find out where the target locations are and surveil them in the right sequence to find the infiltrators.

The actions available to the agent are the motion actions in the cardinal directions as well as a surveillance action, each of which is deterministic in its outcome. Every action taken results in a reward of -1, an unsuccessful surveillance action (i.e. inspecting the target v-location in the wrong order) results in a reward of -10, while a successful surveillance action (i.e. inspecting the correct

Figure 7: CLUSTERING FOR THE WINDY CORRIDOR DOMAIN. The clustering was obtained by running Search-Clustering on the full set of 100 MDPs for this domain. Each point in the 2D grid is a MDP with the goal location and wind probability given by the x and y axis respectively. The colours indicate the cluster to which the MDP was found to belong, and shows that all the MDPs with the same goal state are assigned to the same cluster. Additionally, the cluster centres $\mathcal{M}^i$ are each marked with a $*$.

v-location at the right time) results in a reward of 200. If instead of surveilling the correct v-location, the agent surveils a location *adjacent* to it, then it receives a reward of 190. Each v-location has 3 other v-locations that are adjacent (see Figure 9). Hence two MDPs $\mathcal{M}$, $\mathcal{M}'$, each corresponding to different v-location sequences (v_1, v_2, v_3) and (v'_1, v'_2, v'_3), are similar in terms of their optimal policy/d_V distance, and belong to the same cluster, if each pair of v-locations v_i and v'_i are adjacent – because in this case, the optimal policy of $\mathcal{M}$ will yield near-optimal sequence of rewards of 190s when applied in $\mathcal{M}'$ and vice versa.

We present the following results for experiments run with a combination of different numbers of previous MDPs (referring to the surveillance tasks which have been encountered before, and possibly subsequently clustered) and numbers of target locations. The results show that the more complex the transfer task is, the better EXP-3-Transfer with clustering performs compared to Probabilistic Policy Reuse, where complexity is measured in terms of the number of previous MDPs and the difficulty of the target task.

- We compare the performance of EXP-3-Transfer, Probabilistic Policy Reuse and Q-learning as the complexity of the transfer problem increases. Here, the complexity of the transfer problem is both the number of previous tasks, and the complexity of the MDP itself (i.e. the number of target v-locations). These results are referred to as clustering gains.

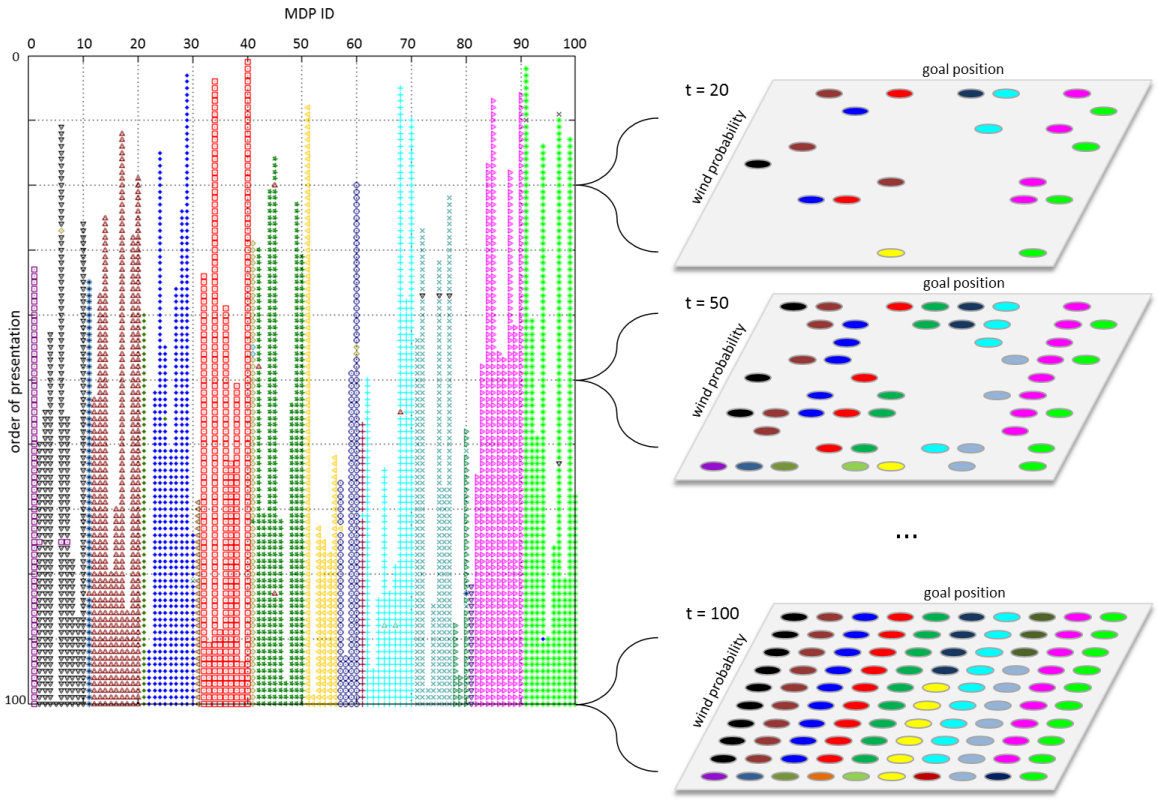


Figure 8: INCREMENTAL CLUSTERING FOR THE WINDY CORRIDOR DOMAIN. The incremental clustering on the left was obtained by running Search-Clustering on an increasing number of MDPs. The 100 MDPs (represented by a value on the x axis) in this domain are presented in a random order to the algorithm, with the y axis showing time of first presentation to the algorithm. All points of the same colour have been assigned to the same cluster. On the right are three zoomed-in time slices from the incremental clustering (at $t = 20$, $t = 50$ and $t = 100$ respectively). Each time slice shows the intuitive interpretation of MDPs in the representation of Figure 7.

- We compare the effect of different types of clusterings (in Table 1) for EXP-3-Transfer with $T = 10,000$. These results are called clustering comparisons.
- We compare the effect of having different $T \in \{1000, 5000, 10000\}$ for EXP-3-Transfer with clustering for various number of previous tasks. These results are the time comparisons.

7.3.1 CLUSTERING GAINS

We first study the effects of clustering, by comparing the performance of both EXP-3-Transfer and Probabilistic Policy Reuse with and without clustering. The results presented in this section summarise those of the complete experiment set, the results of which are provided in Appendix D.

Figure 11 demonstrates the performance of the two algorithms on two different task variants in the surveillance domain: having either a sequence of two or three v-locations. The figure measures the *clustering gain*, being the difference in performance with and without clustering. This figure

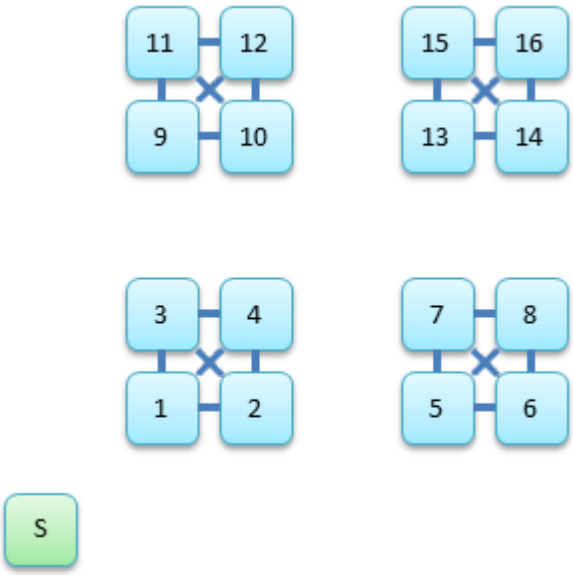


Figure 9: THE SURVEILLANCE DOMAIN. This caricature of the surveillance domain shows 16 surveillance locations (v-locations) marked in blue. The start location is marked S. The full domain used in our experiments is a 48×48 gridworld with 64 v-locations. Each MDP in the domain requires the agent to surveil i different locations, $i \in \{1, 2, 3, 4\}$ in a particular sequence to receive a positive reward of 200 for each location surveilled. Surveilling at a wrong location results a negative reward of -10 (the infiltrators have escaped). Each action taken gives a reward of -1. The target v-locations are clustered spatially into groups of 4 (as shown by the connections in the figure), such that surveilling one location in the cluster instead of the other results in a reward of 190 (a penalty of $200 - 190 = 10$) but does not end the episode (see Figure 10 for further details).

does not show a comparison with Q-learning. In the full results given in Appendix D, we show that EXP-3 consistently outperforms Q-learning by a large margin, indicating that our algorithm escapes negative transfer.

As can be seen, EXP-3-Transfer always benefits from using clustering. Furthermore, the more complex the task, the better the performance. This is observed in the general upwards trend in the curves with an increasing number of previous MDPs, and the fact that the curve for the 3 v-locations lies above the curve for the 2 v-locations. This result is in complete agreement with our expectations, that in a bandit-like algorithm lowering the number of arms will result in lower regret. In addition, this result indicates that our clustering algorithm retains the correct arms so that with the removal of arms, the performance of EXP-3-Transfer is not affected adversely.

Interestingly, for Probabilistic Policy Reuse the trend is reversed. It appears that clustering does not help this algorithm, and the clustering additionally becomes more detrimental as the task complexity increases. Our conjecture regarding the reason for this is that Probabilistic Policy Reuse uses the source policies not as potential optimal policies, but rather as exploration devices. By clustering the MDPs, we remove arms and hence reduce the number of exploration policies, which lowers the scope for exploration. This in turn results in negative performance gain for Probabilistic Policy Reuse.

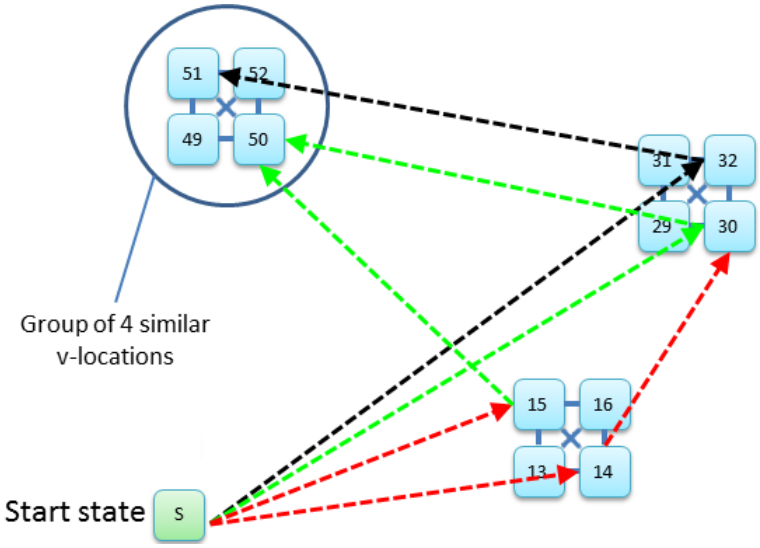


Figure 10: TRAJECTORY EXAMPLES. Example of four different types of trajectories for the surveillance domain, showing three groups of four v-locations. The task MDP requires visiting two v-locations, identified as numbers 30 and 50 in sequence. A green line represents moving to the correct v-location in sequence, yielding a reward of 200. A black line represents moving to an incorrect v-location, which is in the same group as the correct one, for a reward of 190. A red line depicts movement to the incorrect group, for a reward of -10 when the surveillance action is taken.

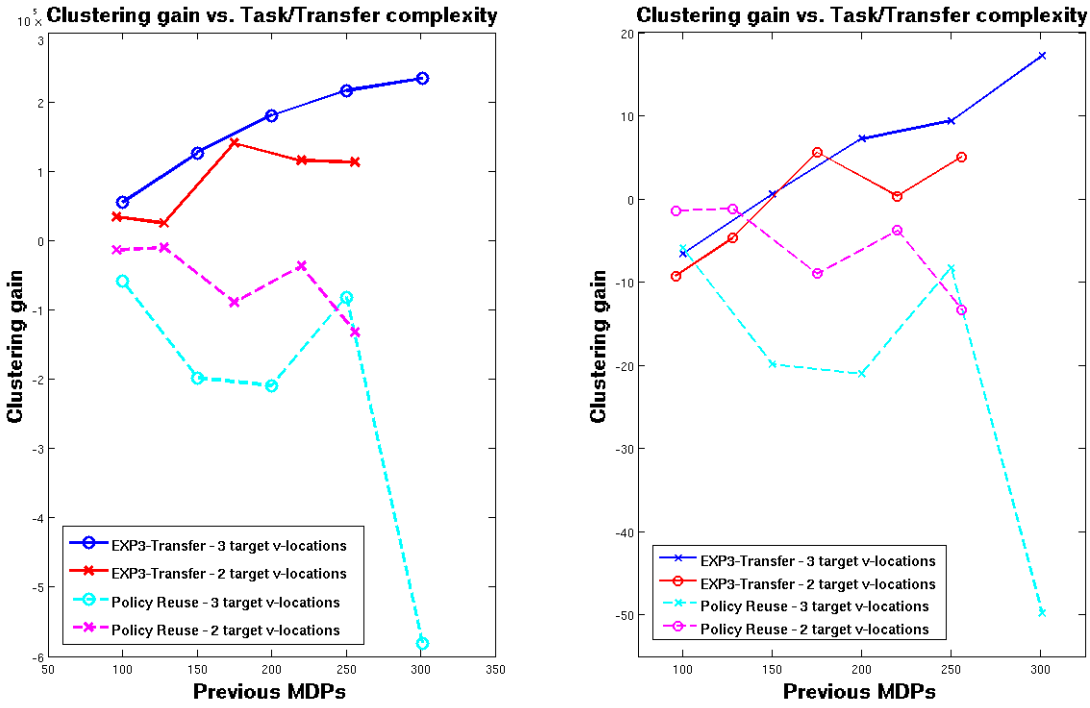
It is also interesting to note that in Figure 11b, which shows the gain in terms of the final reward obtained, the initial gain for EXP-3-Transfer and PPR are both negative, and the gain for PPR is higher. However, as the transfer complexity increases (both in terms of previous MDPs and task complexity) the cumulative reward gain becomes positive for EXP-3-Transfer, while for PPR it continues to decrease.

Given that the above figures show that PPR does not benefit from clustering, we compare the cumulative reward obtained by EXP-3-Transfer with clustering and PPR without clustering for the complex 3-target-v-locations problem in Figure 12. This result shows that EXP-3-Transfer completely dominates PPR, with the difference becoming particularly stark when the number of previous tasks increases to 300.

7.3.2 CLUSTERING COMPARISON

We now compare the performance of EXP-3-Transfer when using the different types of clustering methods reported in Table 1. As in the previous section, we examine the change in performance with increasing complexity of the transfer tasks. The summary of these results is given in Figure 13, and again, the full results are provided in Appendix D.

As can be seen, EXP-3-Transfer using Search-Clustering to obtain the source policies outperforms the case when we do not cluster the previous tasks. This confirms the result reported in the previous section. However, in addition we also observe that transfer after using the greedy clustering scheme performs about as well as using Search-Clustering for the 100 and 260 previous MDPs,



(a) CUMULATIVE CLUSTERING GAINS.

(b) FINAL CLUSTERING GAINS.

Figure 11: CLUSTERING GAINS. The above figures show the **clustering gain** for EXP-3-Transfer and Probabilistic Policy Reuse. For each (x, y) data-point in each curve, the y-value is the difference in performance with and without clustering when there are x previous MDPs. The performance measured in cumulative clustering gains (Figure 11a) is the total cumulative discounted reward over $10,000$ episodes. The performance measure in final clustering gains (Figure 11b) is the discounted reward in the final episode. We again note that each (x, y) point is averaged over 10 different target tasks with 10 trials per task.

but for the 125 previous MDPs, Search-Clusterings is significantly better. Search-Clustering and Greedy Clustering are comparable for those two numbers of previous MDPs largely due to the structure of the domain, where every element of each group of tasks is similar to every other task in the same group. That is, when the agent surveils any v-location in the same group as the true v-location required by the task it receives near-optimal rewards (see Figure 10).

However, greedy clustering would fail on a more complex variant of the surveillance domain. To understand this issue, recall from the beginning of Section that two MDPs $\mathcal{M}$, $\mathcal{M}'$, corresponding to v-location sequences (v_1, v_2, v_3) and (v'_1, v'_2, v'_3), are similar in terms of their optimal policy, and hence belong to the same cluster, if each pair of v-locations v_i and v'_i are adjacent. In this case surveiling v_i instead of v'_i yeilds a reward of 190 in $\mathcal{M}'_i$ and *vice versa*. Hence within each correct cluster of MDPs we have a *symmetry*: the d_V distance between each pair of MDPs is very similar. At the same time the d_V distance to MDPs not in the same cluster is quite different. So if in greedy clustering if we start with any MDP in a given cluster, we will find the other MDPs in the

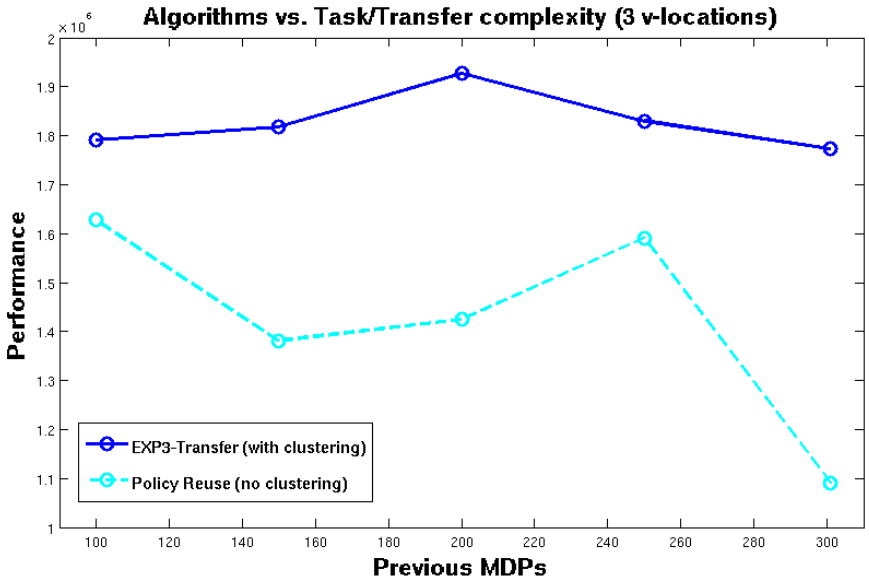


Figure 12: CUMULATIVE REWARD SUMMARY. This figure shows the final cumulative rewards after $10,000$ episodes for EXP-3-Transfer with clustering and Probabilistic Policy Reuse without clustering for the surveillance domain with 3-target-v-locations. The x-axis shows the number of previous MDPs.

cluster. Greedy clustering will no longer work if we break this symmetry by explicitly modifying the domain definition.

One simple way to do this is as follows. In the complex domain, for each correct cluster we have a single MDP (the centroid $\mathcal{M}^i$) that has low d_V distance to all other elements of the cluster, while every other MDP of the cluster has a high d_V distance to the other non-centroid elements. To get this effect, we have 5 v-locations per group of adjacent locations. We then change the reward function so that in each group of adjacent locations $\{v^1, v^2, v^3, v^4, v^5\}$ (see figure 14), the reward is asymmetric - we designate a single v-location, say v^1 which yields a reward of 190 if it is surveilled instead of v^i and vice-versa, but for all the other v-locations, surveilling v^j instead of v^i, $i, j \neq 1$, yields a reward of -10. Hence the $\mathcal{M}^i$ would be the MDP with v-location sequence (v_1^1, v_2^1, v_3^1). In this case, greedy clustering will fail to learn every cluster for which it does not start with the centroid MDP $\mathcal{M}^i$ because the for the non-centroid MDPs, the d_V distance to the other MDPs would be too large. The result for the complex domain is given in table 3. We see that, transfer using the clusters produced by Search-Clustering outperforms transfer using clusters discovered by greedy clustering approach by a factor of 2 as we increase the number of previous MDPs. Recall that each point is obtained from averaging over 10 different target tasks.

7.3.3 TIME COMPARISONS

Finally we examine the effect of the T parameter on performance on the original surveillance domain. Recall that the T parameter affects both the clustering algorithm Search-Clustering and EXP-3-Transfer, and is the time duration in terms of the number of episodes over which the transfer

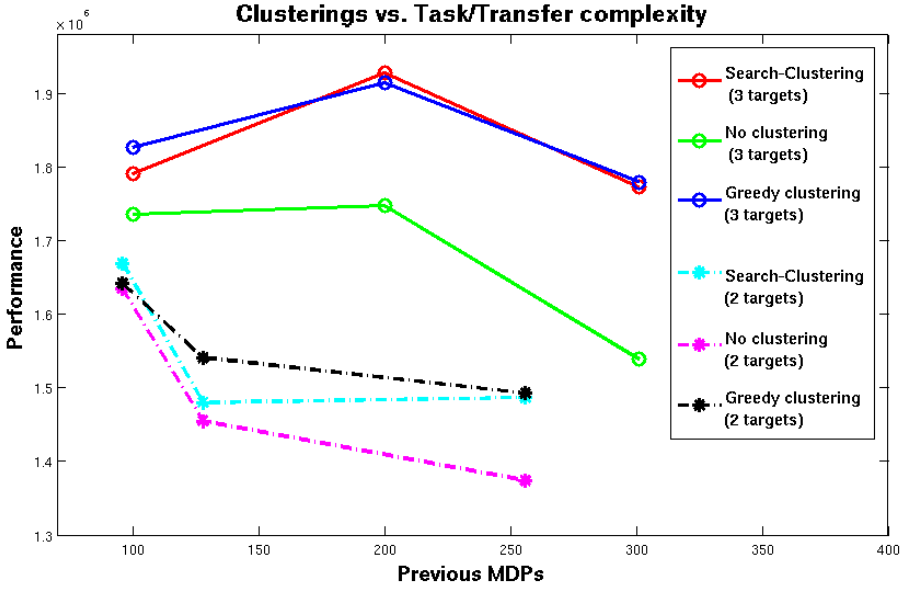


Figure 13: EFFECT OF CLUSTERING METHODS. This figure compares the performance of EXP-3-Transfer run with no clustering, the greedy clustering procedure and Search-Clustering. The performance is measured in terms of the total cumulative discounted rewards over $10,000$ episodes.

No. of Previous MDPs	Search Clustering	Greedy Clustering
100	8.7102×10^6	8.3667×10^6
200	8.3667×10^6	3.809×10^6

Table 3: This table gives the performance of EXP-3-Transfer in the complex surveillance domain when using greedy clustering vs Search-Clusterings for tasks with 2 target v-locations. The performance is measured in terms of the total cumulative discounted rewards over $10,000$ episodes and averaging over 10 different target tasks. The table shows that the performance of Search-Clusterings remains steady while the performance of greedy clustering falters when we have more previous tasks to draw from.

procedure is run. We performed experiments with 7 different combinations of numbers of previous MDPs and MDP complexity. The results of these are all qualitatively similar, and so we present only two graphs in Figure 15 for the most complex and the least complex transfer problem we have considered. We relegate the remaining graphs for the rest of the experiments to Appendix D.2.

As these results show, the performance curve for EXP-3-Transfer with parameter T lies above the curves of EXP-3-Transfer with parameter $T' > T$, for any $t \leq T$. This illustrates the effect of optimising transfer performance for a set time duration.

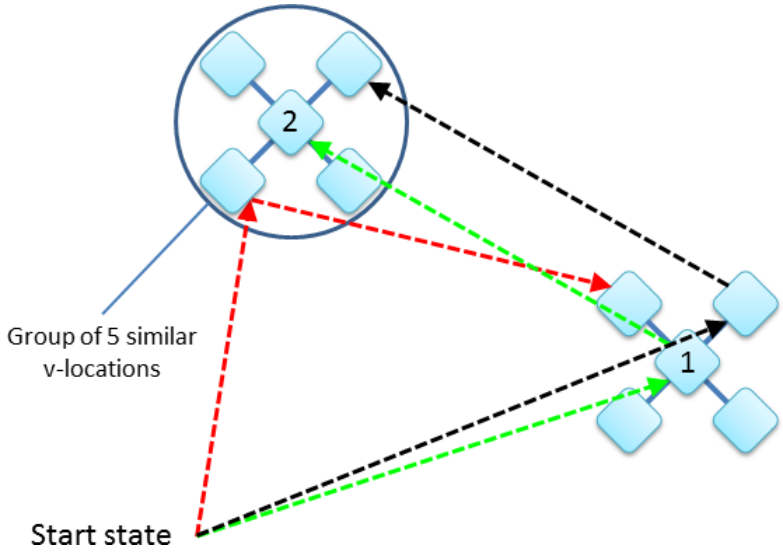


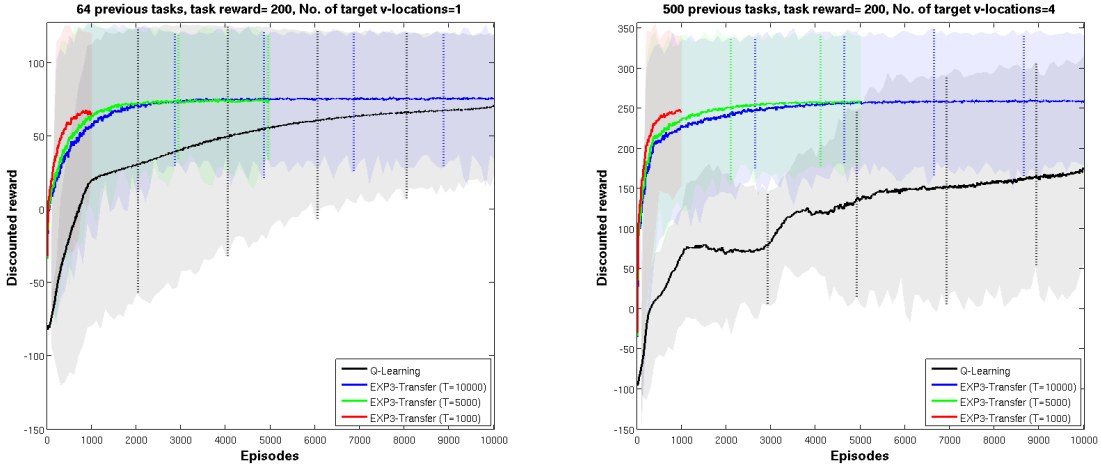
Figure 14: THE COMPLEX GRAPH DOMAIN. The complex graph domain consists of groups of 5 adjacent v-locations. In our experiments we used 16 groups but for simplicity we only show two groups here. Surveilling the central v-location v^1 instead of any of the others has reward 190 and vice-versa, while surveilling v^i instead of v^j, $i, j \neq 1$, has reward -10. As before, each MDP consists of surveilling the correct sequence of target v-locations, depicted here as '1' followed by '2'. As in Figure 10, a green line shows correct movement (reward 200), a black line shows movement to the wrong v-location in the right group (reward 190), and a red line corresponds to moving to the wrong group before surveilling (reward -10).

8. Conclusion

In this paper we developed a framework to concisely represent a large number of previous MDPs by a smaller number of source MDPs for transfer learning. We presented a principled online transfer learning algorithm, a principled way to evaluate source sets for use in this algorithm and way to find the source set. The key idea was to cluster the previous MDPs and then use the representative element of each cluster as the source tasks. We also presented extensive experiments to show the efficacy of our method. We now discuss several interesting directions for future work.

In this paper we only considered discrete domains. However, it is possible to translate the overall approach to the continuous setting. In particular, to apply our approach to continuous space problem, all we will need is a pure RL algorithm (as an arm in EXP-3-Transfer) and a way to evaluate policies (to compute the d_V distances). All our definitions, algorithms and results will then hold true in this setting. This is because our algorithms EXP-3-Transfer, MHAV and Search-Clustering and distance function d_V treats the underlying MDPs and policies as black boxes with certain properties. The discreteness of the MDP is never exploited or required in either the algorithms or their analysis.

Finally, we end by pointing out that the idea of clustering a set of tasks to obtain a representative set is much more general. For instance, any other cost function derived under different assumptions can be applied with the clustering approach. As another example, the clustering approach may also be used in multi-agent systems to group together opponents according to whether the same policy of



(a) LOW COMPLEXITY TRANSFER PROBLEM.

(b) HIGH COMPLEXITY TRANSFER PROBLEM.

Figure 15: EFFECT OF THE T PARAMETER. These figures show the learning curve of EXP-3-Transfer when run for different numbers of time steps (parameter T). This affects both the clustering and the arms chosen by EXP-3-Transfer. The parameters for each experiment are given in the title of the respective figure. As shown, for shorter T, the EXP-3-Transfer run with the lowest $T = 1,000$ is optimal. For the intermediate duration, $T = 5,000$ is optimal, and for the remaining time $T = 10,000$ is optimal. Figures 15a and 15b respectively give the curves for the lowest and highest complexity task that was run. The shaded areas indicate one standard deviation.

ours is equally effective against opponents in the same group. It will also be interesting to implement these methods and algorithms on scaled up, real version of the types of problems considered in this paper. We plan to pursue these and other extensions to the above in future work.

Appendix A. Proofs

Proof [Proof of Theorem 2] A direct application of Corollary 3.2 in (Auer et al., 2002b) is not possible because our algorithm diverges from EXP-3 because the number of arms possibly decreases across time steps. The proof of the first part, while structurally similar tp the proof of Theorem 3.1 of Auer et al. (2002b), is different in some crucial detail due to the removal of arms/policies. The second part, where we deal with arms that were removed, is novel.

Recall that C_t is the set of policies not yet eliminated at the beginning of step t EXP-3-Transfer, and let $c_t = |C_t|$. Let $W_t \triangleq \sum_{i \in C_t} w_i(t)$, and $\tilde{W}_{t+1} \triangleq \sum_{i \in C_t} w_i(t+1)$ (note the C_t, rather than C_{t+1} in the summation in $\tilde{W}_{t+1}$). Then for all sequences of policies $i_1, i_2, \cdots, i_T$, drawn by EXP-3-Transfer,

$$\begin{aligned}
\frac{W_{t+1}}{W_t} &\overset{(1)}{\leq} \frac{\tilde{W}_{t+1}}{W_t} \overset{(2)}{=} \sum_{i \in C_t} \frac{w_i(t+1)}{W_t} \overset{(3)}{=} \sum_{i \in C_t} \frac{w_i(t)}{\sum_{i \in C_t} w_i(t)} \exp[\beta \hat{x}_i(t)/(c+1)] \\
&\overset{(4)}{=} \sum_{i \in C_t} \frac{p_i(t) - \beta/c_t}{1-\beta} \exp[\beta \hat{x}_i(t)/(c+1)] \overset{(5)}{=} \sum_{i \in C_t} \frac{p_i(t) - \beta/c_t}{1-\beta} \exp[\beta \hat{x}_i(t)/c_t] \\
&\overset{(6)}{\leq} 1 + \frac{\beta/c_t}{1-\beta} x_{i_t}(t) + \frac{(e-2)(\beta/c_t)^2}{1-\beta} \sum_{i \in C_t} \hat{x}_i(t)
\end{aligned} \tag{27}$$

In the above, $^{(1)}$ follows because $\tilde{W}_{t+1} \geq W_{t+1}$ as $|C_t| \geq |C_{t+1}|$, and because all the weights w_{t+1} are positive. $^{(2)}$ follows by the update equation for $w_i(t+1)$ in line 9 of EXP-3-Transfer. $^{(3)}$ follows by definition of W_t. $^{(4)}$ follows by definition of $p_i(t)$ in line 4 of EXP-3-Transfer. $^{(5)}$ holds because $c_t \leq c+1$ and because the term in the exponential is positive. Finally, $^{(6)}$ follows by the identical reasoning used to derive (8) in the proof of Theorem 3.1 (Auer et al., 2002a).

Now, we can proceed along similar lines as in Theorem 3.1. Using the fact that $1 + x \leq \exp(x)$ gives us

$$\ln \frac{W_{t+1}}{W_t} \leq \frac{\beta/c_t}{1-\beta} x_{i_t}(t) + \frac{(e-2)(\beta/c_t)^2}{1-\beta} \sum_{i \in C_t} \hat{x}_i(t)$$

Now since c_t is non-increasing, $\beta/c_t \leq \beta/c_T$, and so

$$\ln \frac{W_{t+1}}{W_t} \leq \frac{\beta/c_T}{1-\beta} x_{i_t}(t) + \frac{(e-2)(\beta/c_T)^2}{1-\beta} \sum_{i \in C_t} \hat{x}_i(t)$$

Summing over t telescopes and gives us

$$\ln \frac{W_T}{W_1} \leq \frac{\beta/c_T}{1-\beta} \sum_{t=1}^{T} x_{i_t}(t) + \frac{(e-2)(\beta/c_T)^2}{1-\beta} \sum_{t=1}^{T} \sum_{i \in C_t} \hat{x}_i(t)$$

Now, going in the opposite direction, for each $\rho_j \in C_T$ we have

$$\ln \frac{W_T}{W_1} \geq \ln \frac{w_j(T)}{c+1} = \frac{\beta}{c_T} \sum_{t=1}^{T} \hat{x}_j(t) - \ln(c+1)$$

Putting these together, we have for each $\rho_j \in C_T$,

$$\sum_{t=1}^{T} x_{i_t}(t) \geq (1-\beta) \sum_{t=1}^{T} \hat{x}_j(t) - \frac{c_T \ln(c+1)}{\beta} - \frac{(e-2)\beta}{c_T} \sum_{t=1}^{T} \sum_{i \in C_t} \hat{x}_i(t) \tag{28}$$

Rearranging, we get

$$\sum_{t=1}^{T} \hat{x}_j(t) - \sum_{t=1}^{T} x_{i_t}(t) \leq \beta \sum_{t=1}^{T} \hat{x}_j(t) + \frac{c_T \ln(c+1)}{\beta} + \frac{(e-2)\beta}{c_T} \sum_{t=1}^{T} \sum_{i \in C_t} \hat{x}_i(t) \tag{29}$$

Taking expectation in terms of the randomization in the algorithm in both sides above, conditional of $\rho_j \in C_T$, we have

$$\sum_{t=1}^{T} x_j(t) - \mathbb{E}_{E3T}[\sum_{t=1}^{T} x_{i_t}(t) | \rho_j \in C_T] \leq \beta \sum_{t=1}^{T} x_j(t) + \frac{c_T \ln(c+1)}{\beta} + \frac{(e-2)\beta}{c_T} \sum_{t=1}^{T} \sum_{i \in C_t} x_i(t) \tag{30}$$

For the final term on the right hand side, we used the fact $\mathbb{E}[\hat{x}_i(t) | i_1, i_2, \cdots, i_{t-1}, \rho_j \in C_T] \leq x_i(t)$ for any i. To see this, note that there are two possibilities – either $\rho_i \in C_t$, or $\rho_i \notin C_t$. If $\rho_i \in C_t$ then $\mathbb{E}[\hat{x}_i(t) | i_1, i_2, \cdots, i_{t-1}, \rho_j \in C_T] = x_i(t) \geq 0$, and otherwise, $\mathbb{E}[\hat{x}_i(t) | i_1, i_2, \cdots, i_{t-1}, \rho_j \in C_T] = 0$ as $p_i(t) = 0$.

Now, using the fact that $\sum_{t=1}^{T} \sum_{i \in C_T} x_i(t) \leq c_T T$ (as $x_i(t) \in [0,1]$) and then rearranging, we get for each $\rho_j \in C_T$,

$$\begin{aligned}\sum_{t=1}^{T} x_j(t) - \mathbb{E}_{E3T}[\sum_{t=1}^{T} x_{i_t}(t) | \rho_j \in C_T] &\leq \beta \sum_{t=1}^{T} x_j(t) + c_T \ln(c+1) + (e-2)\beta T \\ &= \frac{c_T \ln(c+1)}{\beta} + (e-1)\beta T\end{aligned}$$

Plugging in the value of β in the theorem statement we get

$$\sum_{t=1}^{T} x_j(t) - \mathbb{E}_{E3T}[\sum_{t=1}^{T} x_{i_t}(t) | \rho_j \in C_T] \leq \left(\frac{|C_T|}{\sqrt{c+1}} + \sqrt{c+1} \right) \sqrt{(e-1)\ln(c+1)/T} \tag{31}$$

Now note that $\Delta R (1-\gamma)^{-1} \mathbb{E}_{P,R}[\sum_{t=1}^{T} x_j(t)] = T V^{\rho_j}$, so taking expectation with respect to P, R, dividing by T and multiplying by $\Delta R (1-\gamma)^{-1}$ we get from (31)

$$V^{\rho_j} - \frac{1}{T} \mathbb{E}_{E3T}[\sum_{t=1}^{T} \bar{x}_{i_t}(t) | \rho_j \in C_T] \leq \frac{\Delta R}{1-\gamma} \left(\frac{|C_T|}{\sqrt{c+1}} + \sqrt{c+1} \right) \sqrt{(e-1)\ln(c+1)/T}$$

This completes the proof of the first part.

To prove the second part, we need the Hoeffding bound (see, for instance, (Dubhashi and Panconesi, 2009) for an exposition) which states that if $y_{1:n} \triangleq y_1, y_2, \cdots, y_n$ are i.i.d. draws of a random variable Y, with $Y_i \in [a,b]$, and $\bar{y}_n$ is the empirical mean of the y_i, then

$$Pr[|\bar{y}_n - \mathbb{E}(Y)| > \epsilon] \leq \exp[-2n\epsilon^2/(b-a)^2] \tag{32}$$

In the sequel, we will assume that $b = 1, a = 0$, and so the denominator in the exponent on the left hand side of the equation is just 1. This bound then has the following simple and well known consequence. Assume we have two i.i.d. samples $y_{1:n}$ and $y'_{1:m}$, drawn from two random variables Y and Y'. Assume that $\bar{y}_n - \bar{y'}_m > \epsilon$, and n and m both satisfy $\exp[-2n\epsilon^2/4] \leq \delta'/2$ and $\exp[-2m\epsilon^2/4] \leq \delta'/2$. Then, by (32)

$$Pr[|\bar{y}_n - \mathbb{E}(Y)| > \epsilon/2] \leq \delta'/2, \qquad Pr[|\bar{y'}_n - \mathbb{E}(Y')| > \epsilon/2] \leq \delta'/2 \tag{33}$$

Then, by the triangle inequality and the union bound, with probability at least $1 - \delta'$, $\mathbb{E}(Y) > \mathbb{E}(Y')$.

Now in line 13 of EXP-3-Transfer, we remove a source policy arm if $\epsilon = z_j/n_j - z_k/n_k$, we have $\epsilon/2 > \sqrt{-\ln(\delta/2c)(2n_j)^{-1}}$ and $\epsilon/2 > \sqrt{-\ln(\delta/2c)(2n_k)^{-1}}$. This implies, that with probability $> 1 - \delta/c$, $V^{\rho_j} > V^{\rho_k}$. Since there are c arms, this implies that if there is an arm that was removed, by the union bound with probability at least $> 1 - \delta$, $V^{\rho_j} > V^{\rho_k}$ for some arm j for every arm k that is eventually removed.

■

For the next two proofs, we need to restate Lemma 1 in (Strehl and Littman, 2008).

Lemma 19 *[Strehl, Li and Littman] Let the new $\mathcal{M}_{N+1}$ have transition and reward functions R_{N+1} and T_{N+1}. Let $|R_i(s,a) - R_{N+1}(s,a)| \leq K_i$ and $|T_i(.|s,a) - T_{N+1}(.|s,a)| \leq K'_i$ where R_i and T_i are the reward and transition functions for MDP $\mathcal{M}_i$. Then, for any policy π and state s,*

$$|V^{\pi}_{N+1}(s) - V^{\pi}_{i}(s)| \leq K(i) \tag{34}$$

where $K(i) \triangleq \frac{K_i + \gamma K'_i R_{max}}{(1-\gamma)^2}$ (first defined in (10)). □

Now we can state our proof.

Proof [Proof of Lemma 5] Fix any previous $\mathcal{M}_k$ and let $\mathcal{M}^i$ be the centroid of the cluster A_i of $\mathbf{A}$ such that $\mathcal{M}_k \in A_i$. By definition, the optimal policy of $\mathcal{M}^i$, used as an arm in EXP-3-Transfer, is ρ_i and $V_k^{\pi_k^*} - V_k^{\rho_i} \leq \epsilon_i$. By Lemma 19, $|V_{N+1}^{\pi_k^*} - V_k^{\pi_k^*}| \leq K(k)$ and $|V_{N+1}^{\rho_i} - V_k^{\rho_i}| \leq K(k)$. Putting the three inequalities together, we have $|V_{N+1}^{\pi_k^*} - V_{N+1}^{\rho_i}| \leq \epsilon_i + 2K(k)$. By Theorem 2, if ρ_i was not eliminated by EXP-3-Transfer, $V_{N+1}^{\rho_i} - \mathbb{E}[G_{E3T}]/T \leq g(c)$, and therefore $V_{N+1}^{\pi_k^*} - \mathbb{E}[G_{E3T}]/T \leq g(c) + \epsilon_i + 2K(k)$.

Now consider the case where the policy ρ_i was eliminated at some point. Then, by the second part of Theorem 2, there exists a policy $\rho_{i'}$ for which with probability at least $1 - \delta$, $V^{\rho_i} \leq V_{N+1}^{\rho_{i'}}$. From the relationship between ρ_i and π_k^* established above, we get that with probability $1 - \delta$, $V_{N+1}^{\pi_k^*} \leq V_{N+1}^{\rho_{i'}} + \epsilon_i + 2K(k)$.

■

Proof [Proof of Theorem 9] First, let $|V| = M$, and given any ordering of the elements of V, identify each vertex $v \in V$ with its position in the ordering – so we can take $V = \{1, 2, \cdots, M\}$. Let $\mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_M$ be a set of MDPs defined on a state space $\mathcal{S} = \{s\}$, and action space $\mathcal{A} = \{1, 2, \cdots, M\}$. The transition function for the MDPs in this case is trivial (all actions transition with probability 1 from s to s). The reward function for MDP i defined as follows. $R_i(s, a_i) = 0$; if

$(i,j) \in E$ then $R_i(s,a_j) = 0$, otherwise $R_i(s,a_j) = -hMg(M)$ where $h > 1$ and g is the function used in Definition 6 to define the cost function for clusters. In the following we will identify MDP $\mathcal{M}_i$ with vertex $i \in V$ and this way show that the optimal clustering for this corresponds to a maximal clique under the mapping $i \to v_i$ and $A \to V_i$.

By construction, $\pi_i^*(s) = a_i$, $V_i^* = 0$, $V_i^{\pi_j^*} = V_i^{a_j} = 0$ iff $(i,j) \in E$, and $V_i^{a_j} = -hMg(M)$ otherwise. Hence, by the definition in (7),

$$d_V(\mathcal{M}_i, \mathcal{M}_j) = \begin{cases} 0 & \text{iff } (i,j) \in E \\ hMg(M) & \text{otherwise} \end{cases} \tag{35}$$

Now recall that $cost_m \triangleq g(c) + \bar{\epsilon}_m$. Let an optimal clustering be $\mathbf{A}^*$ and let $A(\mathcal{M}_i)$ denote the cluster in $\mathbf{A}^*$ that $\mathcal{M}_i$ belongs to. We now show that if $\mathcal{M}_{i_1}, \mathcal{M}_{i_1}, \cdots, \mathcal{M}_{i_l} \in A \in \mathbf{A}^*$, then $i_1, i_2, \cdots, i_l$ form a clique in G. In other words, we show that, if $A(\mathcal{M}_i) = A(\mathcal{M}_j)$ then $(i,j) \in E$, or equivalently $(i,j) \notin E$ then $A(\mathcal{M}_i) \neq A(\mathcal{M}_j)$. By way of contradiction, assume that $(i,j) \notin E$ but $A(\mathcal{M}_i) = A(\mathcal{M}_j)$. Since i, j do not have an edge between them, by (35) the diameter of $\mathbf{A}^*$ is at least $d_V(\mathcal{M}_i, \mathcal{M}_j) = hMg(M)/M = hg(M)$. Which in turn implies that $cost_m(\mathbf{A}^*) = g(|\mathbf{A}^*|) + hg(M)$. Now consider the clustering $\mathbf{A}'$ obtained by putting each MDP $\mathcal{M}_i$ in its own cluster. This clustering has cost $g(M) + 0 < g(|\mathbf{A}^*|) + hg(M)$ – contradicting the optimality of $\mathbf{A}^*$. Hence, the clusters of $\mathbf{A}^*$ has cost $g(|\mathbf{A}^*|)$ and corresponds to a collection of cliques that partition V – denote this collection of cliques by J^*.

Now note that each collection of cliques $V_1, V_2, \cdots V_j$ that partition V correspond to a clustering $\mathbf{A}$ such that $\mathcal{M}_i, \mathcal{M}_j \in A$ iff $i, j \in V_l$ for some l; in this case $j = |\mathbf{A}|$. Now assume that there is a collection of cliques I such that $|I| < |J^*|$ and let the corresponding clustering be $\mathbf{A}_I$. Then we show that $cost_m(\mathbf{A}_I) < cost_m(\mathbf{A}^*)$, resulting in a contradiction. To see this note that each $\mathcal{M}_i, \mathcal{M}_j \in A \in \mathbf{A}_I$ then $d_V(\mathcal{M}_i, \mathcal{M}_j) = 0$ by (35). Hence the diameter of $\mathbf{A}_I$ = 0. So the cost of $\mathbf{A}_I$ is $g(|\mathbf{A}_I|) + 0 < g(|\mathbf{A}^*|)$ since by definition of $\mathbf{A}_I$, $|\mathbf{A}_I| < |\mathbf{A}^*|$.

Because of the contradiction, J^* is indeed a minimum clique cover, showing that the problem of minimum clique cover can be reduced to the problem of finding the optimal clustering. To complete the proof, we need to show that this reduction takes polynomial time. The only cost in computing a $\mathcal{M}_i$ is setting the reward function, which takes time $C|V|$ for some constant C. ■

Proof [Proof of Lemma 13] To show irreducibility we have to show that for any (λ, y) and (λ', y') there exists a n such that $\bar{P}^n_{\text{MH}}[\lambda', y'|\lambda, y] > 0$. To see this, first note that ϕ_Y was assumed to be irreducible. So, there exists a n_1 such that with $\phi^n_Y(y'|y) > 0$. Now consider a particular path $\mathbf{y} \triangleq yy_1y_2 \cdots y_{n-1}y'$ with probability > 0 under ϕ_Y. From the definition in (21), the probability under $\bar{P}_{\text{MH}}$ of each transition $y_i \to y_{i+1}$ is

$$\beta\bar{\phi}[\lambda, y_{i+1}|\lambda, y_i]\bar{\mathsf{Acc}}_{\lambda,y_i}[\lambda, y_{i+1}] > \beta b\phi_Y[\lambda, y_{i+1}|\lambda, y_i]$$

where the inequality follows as $\bar{\mathsf{Acc}}_\cdot[\cdot] > b$ by assumption in the Lemma statement. Hence, the total probability of the path $yy_1y_2 \cdots y_{n-1}y'$ under $\bar{P}_{\text{MH}}$ is lower bounded by $b^n\beta^n\phi_Y(\mathbf{y})$ (where $\phi(\mathbf{y}) = \prod_{i=0}^n \phi(y_{i+1}|y_i)$). Summing over all possible paths of length n going from y to y' gives that the probability of each (λ, y') from (λ, y) is lower bounded by $b^n\beta^n\phi^n_Y(y'|y) > 0$.

Now assume that $\lambda = \lambda_k$ while $\lambda' = \lambda_{k'}$. If $k < k'$, we can bound the probability under $\bar{P}_{\text{MH}}$ of going from (λ_i, y') to (λ_{i+1}, y'), where $k \leq i < k'$, by $z_i \triangleq \alpha\alpha' \frac{(1-\alpha)}{\alpha}(\lambda_{i+1}/\lambda_i)^{-f(y)}$ (this

follows from definition of $\bar{P}_{\text{MH}}$ and $\bar{\text{Acc}}$). Hence we reach (λ', y') from (λ, y') with probability $z \triangleq \prod_{i=k}^{k'-1} z_i$. By a symmetric argument, if $k' < k$, we reach λ' from λ with probability at least $z' \triangleq \prod_{i=k}^{k'-1} {z'}_i$, where $z'_i \triangleq \alpha(1-\alpha')\frac{\alpha}{(1-\alpha)}(\lambda_i/\lambda_{i+1})^{-f(y)}$. Both z, z' are positive by the finiteness of $f(y)$ and λ_is. Putting all the above together, we have that the probability of transitioning from (λ, y) to (λ', y') is lower bounded by

$$b^n \beta^n \phi_Y^n(y'|y) \min\{z, z'\} > 0$$

which shows that $\bar{P}_{\text{MH}}$ is irreducible.

To show that $\bar{P}_{\text{MH}}$ is aperiodic, it is sufficient to note that $\alpha + \beta < 1$. Then, with probability $1 - \alpha - \beta$, $\bar{P}_{\text{MH}}$ returns to the same state in 1 step, which ensures that the g.c.d. of the set of time steps where $\bar{P}_{\text{MH}}$ returns to the same state is 1. ■

Proof [Proof of Theorem 14] $\bar{\phi}$ is irreducible by Lemma 13 and by construction of an MH chain, $\bar{P}_{\text{MH}}$ has $\bar{\Pi}$ stationary distribution. Hence, by the first part of Theorem 11 $\bar{P}_{\text{MH}}$ converges to $\bar{\Pi}$ in total variation. By the second part of the same theorem, if $\|\bar{P}^t_{\text{MH}}(\cdot|x_0) - \bar{\Pi}(\cdot)\|_{\text{TV}} \leq k$, then for all $t' > t$, $\|\bar{P}^{t'}_{\text{MH}}(\cdot|x_0) - \bar{\Pi}(\cdot)\|_{\text{TV}} \leq k$. ■

Proof [Proof of Theorem 15] As we mentioned above, this proof follows very closely the proof of Theorem 4.9 in (Levin et al., 2009). To begin with, first we note that by irreducibility of $\bar{P}_{\text{MH}}$, the diameter D (defined in (25)) is finite. Hence, by definition of δ in (26), for each x, x' we have that $\bar{P}_{\text{MH}}(x'|x) \geq \delta \bar{\Pi}(x')$.

Let $\bar{M}_{\text{MH}}$ denote the transition matrix for the kernel $\bar{P}_{\text{MH}}$ so that $\bar{M}_{\text{MH}}(x, x') = \bar{P}_{\text{MH}}(x'|x)$ – i.e. row i contains the distribution $\bar{P}_{\text{MH}}(\cdot|x_i)$. Let $\bar{\mathbf{\Pi}}$ denote the transition matrix where each row is $\bar{\Pi}$. Then, setting $\eta \triangleq (1 - \delta)$, we can write

$$\bar{M}_{\text{MH}} = (1 - \eta)\bar{\mathbf{\Pi}} + \eta Q$$

where Q is another transition matrix. To see that Q is a valid transition matrix, note that row i of Q is given by $\eta^{-1}[\bar{P}_{\text{MH}}(\cdot|x_i) - (1 - \eta)\bar{\Pi}(\cdot)]$. Summing the elements of this row, we get $\sum_{x'} \bar{P}_{\text{MH}}(x'|x_i) - (1 - \eta)\bar{\Pi}(x') = \eta$, whence each row of Q sums to 1. Furthermore, by the definition that $(1 - \eta) = \delta$, each entry is also positive, showing that Q is indeed a valid transition matrix.

Now note that for any transition matrix $\mathbf{M}$, $\mathbf{M}\bar{\mathbf{\Pi}} = \bar{\mathbf{\Pi}}$. Additionally, since $\bar{\Pi}$ is stationary for $\bar{P}_{\text{MH}}$, $\bar{\mathbf{\Pi}}\bar{M}_{\text{MH}} = \bar{\mathbf{\Pi}}$. We will now use the above facts to show by induction on k that

$$\bar{M}^{Dk}_{\text{MH}} = (1 - \eta^k)\bar{\mathbf{\Pi}} + \eta^k Q^k \tag{36}$$

which will imply the convergence we seek.

Clearly (36) is true for $k = 0$. Assume, as the inductive hypothesis, that it is true for $k \leq n$. Then, we have

$$\begin{aligned}
\bar{M}^{D(n+1)}_{\text{MH}} &= \bar{M}^{Dn}_{\text{MH}} \bar{P}^D_{\text{MH}} \\
&= (1 - \eta^n)\bar{\mathbf{\Pi}} + \eta^n Q^n \bar{M}^D_{\text{MH}} \\
&= (1 - \eta^n)\bar{\mathbf{\Pi}} + \eta^n Q^n [(1 - \eta)\bar{\mathbf{\Pi}} + \eta Q] \\
&= (1 - \eta^n)\bar{\mathbf{\Pi}} - \eta^{n+1}\bar{\mathbf{\Pi}} + \eta^n \bar{\mathbf{\Pi}} + \eta^{n+1} Q^{n+1} \\
&= (1 - \eta^{n+1})\bar{\mathbf{\Pi}} + \eta^{n+1} Q^{n+1}
\end{aligned}$$

The first equality is just the definition of k-step transitions. The second equality is obtained by applying the inductive hypothesis and because $\bar{\mathbf{\Pi}}\bar{M}_{\mathrm{MH}} = \bar{\mathbf{\Pi}}$. The third and fourth equality follows from applying the inductive hypothesis on $\bar{M}^D_{\mathrm{MH}}$ and the two facts about $\bar{\mathbf{\Pi}}$ established above. The final equality is obtained by cancelling out the terms.

Now $\eta^k \to 0$ as $k \to \infty$, and so each row of $\bar{M}_{\mathrm{MH}}$ converges to $\bar{\Pi}$. In other words for each x, $\lim_{t\to\infty} \sum_{x'} \bar{P}^t_{\mathrm{MH}}(x'|x) - \bar{\Pi}(x') = 0$. This implies $\lim_{t\to\infty} \|\bar{P}^t_{\mathrm{MH}}(\cdot|x) - \bar{\Pi}(\cdot)\|_1 = 0$. Now since $\|\bar{P}^t_{\mathrm{MH}}(\cdot|x) - \bar{\Pi}(\cdot)\|_{\mathrm{TV}} = \frac{1}{2}\|\bar{P}^t_{\mathrm{MH}}(\cdot|x) - \bar{\Pi}(\cdot)\|_1$, this completes the proof. ■

Proof [Proof of Lemma 16] Fix any two (λ, y) and (λ', y') and let $\mathbf{x} \triangleq x_0 x_1 \cdots x_n$ be a path with $x_0 = (\lambda, y)$ and $x_n = (\lambda', y')$. Assume that this path has positive probability under $\bar{P}_{\mathrm{MH}}$ for certain value a, b, c, respectively of α', α, β. Then, by definition (23) of $\bar{P}_{\mathrm{MH}}$, the probability of this path has the form $Ca^k(1-a)^{k_2}b^{k_2}c^{k_3}(1-b-c)^{k_4}$ where the k_i are integers and C is a constant. Then, under a difference set of values a', b', c', the probability of this path has the form $Ca'^k(1-a')^{k_2}b'^{k_2}c'^{k_3}(1-b'-c')^{k_4}$. Since $\alpha', \alpha, \beta \in (0,1)$, this probability must also be non-zero. Hence the set of paths of positive probability are invariant with respect to the values of α', α and β. Since D is the length of the shortest path of positive probability, this proves the lemma. ■

Proof [Proof of Lemma 18] We just need to show that, for any two clusterings $\mathbf{A}$ and $\mathbf{A}'$, only a finite number of re-arrangement steps is sufficient to obtain $\mathbf{A}'$ from $\mathbf{A}$. Let the clusters of $\mathbf{A}'$, in some order, be $A'_1, A'_2, \cdots, A'_n$. Assume that the points of A'_i are spread across $A_{i_1}, \cdots, A_{i_k}$ with $n_1, n_2, \cdots, n_k$ points respectively. Then, with non-zero probability A'_i will be created with n_1 points from A_{i_1} (see Appendix C for the explicit computation). And from then on, with non-zero probability (again, see the computations given) the points of A'_i in A_{i_j} will be added to A'_i. Hence with non-zero probability A'_i will be created. This holds for each A'_i, and hence we have a non-zero probability of constructing $\mathbf{A}'$ from $\mathbf{A}$. ■

Appendix B. Worst Case Quantification of the Cost Function

This appendix continues from Section 4.2 where we derived the case cost function for a clustering by considering an average case scenario. We now derive a cost function in the worst case setting. To begin, we define the following worst case measure of homogenity of the clustering.

$$\epsilon = \max_i \epsilon_i \tag{37}$$

The main result is as follows.

Lemma 20 *If EXP-3-Transfer is run with source policies derived from* $\mathbf{A}$ *using definition 4 with* β *set as in Theorem 2, we have for each* $1 \le i \le N$*,*

$$\mathbb{E}[V^{\pi^*_i}_{N+1}] - \mathbb{E}[G_{E3T}]/T \le g(c) + \epsilon + \max_i 2K(i)$$

Here the expectation is taken over the randomization of the task drawing process, randomization in EXP-3-Transfer and P_{N+1} *and* R_{N+1} *(same as in Theorem 2).*

Proof By the exact same steps as in the proof of Lemma 5, we have $V^{\pi_k^*}_{N+1} - \mathbb{E}[G_{E3T}]/T \leq g(c) + \epsilon_{k_i} + 2K(k)$. Taking the max over k yields

$$\mathbb{E}[V^{\pi_k^*}_{N+1}] - \mathbb{E}[G_{E3T}]/T \leq g(c) + \epsilon + \max_i 2K(i)$$

which completes the proof. ■

We now discuss which of the two quantifications, worst or average case, is more appropriate. If we have reason to strongly believe that the next MDP $\mathcal{M}_{N+1}$ is chosen by nature adversarially with respect to our choice of cluster $\mathbf{A}$ – that is, nature chooses $\mathcal{M}_{N+1}$ to maximize $\max_{A_j \in \mathbf{A}} \max_{\mathcal{M}_i \in A_j} \epsilon_j + K(i)$ – then clearly, the worst case quantification is the correct way to evaluate a clustering. On the other hand, if we do not have any reason to believe this, then the average case might be more appropriate. For instance a consider a clustering of 1000 MDPs that contains all the MDPs in a 5 clusters, such that 4 of the clusters have diameter < 10 while the 5^{th} one is sparse but wide (say 10 elements with diameter 100). For many domains, we would consider this a good clustering and for this situation, the average case quantification would be $\leq 0.999 \times 10 + 0.001 \times 100 \max_i K(i) \leq 10 + \max_i K(i)$, whereas the worst case quantification would be $100 + \max_i K(i)$. Intuitively this seems a little too pessimistic and indeed, we also observed similar results for our experiments, in the sense that the worst case quantification failed to uncover clusters that we would intuitively consider good. Hence, for the rest of the paper we use the average case quantification to define our distance function.

Appendix C. Computations

Here we present the computation of the ratio $\frac{\phi[\lambda, \mathbf{A}|\lambda', \mathbf{A}']}{\phi[\lambda', \mathbf{A}'|\lambda, \mathbf{A}]}$ defined using (22) and constructed using $\bar{\phi}^M_Y$ defined in Section 5.7. For this section, we set $|\mathbf{A}| = N$. We have four cases to consider.

Case 1: With probability $\alpha\alpha'$, λ' increased and $\mathbf{A}' = \mathbf{A}$. In this case, we have $\phi[\lambda', \mathbf{A}|\lambda, \mathbf{A}] = \alpha\alpha'$, $\phi[\lambda, \mathbf{A}|\lambda', \mathbf{A}] = \alpha(1-\alpha')$ and

$$\frac{\phi[\lambda, \mathbf{A}|\lambda', \mathbf{A}]}{\phi[\lambda', \mathbf{A}|\lambda, \mathbf{A}]} = \frac{1-\alpha'}{\alpha'}.$$

Case 2: With probability $\alpha(1-\alpha')$, λ' decreased and $\mathbf{A}' = \mathbf{A}$. In this case we have $\phi[\lambda', \mathbf{A}|\lambda, \mathbf{A}] = \alpha(1-\alpha')$, $\phi[\lambda, \mathbf{A})|\lambda', \mathbf{A}] = \alpha\alpha'$,

$$\frac{\phi[\lambda, \mathbf{A}|\lambda', \mathbf{A}]}{\phi[\lambda', \mathbf{A}|\lambda, \mathbf{A}]} = \frac{\alpha'}{1-\alpha'}$$

Case 3: With probability $1-\alpha-\beta$, $\lambda' = \lambda$ and $\mathbf{A} = \mathbf{A}'$. $\phi[\lambda, \mathbf{A}, \lambda', \mathbf{A}] = \phi[\lambda', \mathbf{A}, \lambda, \mathbf{A}] = 1-\alpha-\beta$

$$\frac{\phi[\lambda, \mathbf{A}|\lambda, \mathbf{A}]}{\phi[\lambda, \mathbf{A}|\lambda, \mathbf{A}]} = 1$$

Case 4: With probability $\beta\beta'$, $\lambda' = \lambda$ and is rearranged. Now the probability of moving k_i points from A_i to A_j is,

$$P(A_i, A_j; k_i) = N^{-2} PE(k_i; |A_i|, \theta_1) \binom{|A_i|}{k_i}^{-1}$$

The reverse probability now depends on what actually has been moved. We have 4 subcases:

Case 4.1: If k_i points are moved between clusters A_i and A_j from clustering , with $0 < k_i < |A_i|$:

$$P(A_j, A_i; k_i) = N^{-2} PE(k_i; |A_j| + k_i, \theta_1) \binom{|A_j| + k_i}{k_i}^{-1}$$

Now, we have that $\phi[\lambda, \mathbf{A}'|\lambda, \mathbf{A}] = \beta\beta' P(A_i, A_j; k_i)$ and $\phi[\lambda, \mathbf{A}|\lambda, \mathbf{A}'] = \beta\beta' P(A_j, A_i; k_i)$, so that, we have:

$$\frac{\phi[\lambda, \mathbf{A}|\lambda, \mathbf{A}']}{\phi[\lambda, \mathbf{A}'|\lambda, \mathbf{A}]} = \frac{PE(k_i; |A_j| + k_i, \theta_1)\binom{|A_i|}{k_i}}{PE(k_i; |A_i|, \theta_1)\binom{|A_j|+k_i}{k_i}}$$

Case 4.2: If k_i points are moved from cluster A_i to a new cluster $A_{|A|+1}$, with $0 < k_i < |A_i|$:

$$P(A_{N+1}, A_i; k_i) = (N+1)^{-2} PE(k_i; k_i, \theta_1)$$

note that $|A_{N+1}| = k_i$. Now, we have that $\phi[\lambda, \mathbf{A}'|\lambda, \mathbf{A}] = \beta\beta' P(A_i, A_{N+1}; k_i)$ and $\phi[\lambda, \mathbf{A}|\lambda, \mathbf{A}'] = \beta\beta' P(A_{N+1}, A_i; k_i)$. So the desired ratio is:

$$\frac{\phi[\lambda, \mathbf{A}|\lambda, \mathbf{A}']}{\phi[\lambda, \mathbf{A}'|\lambda, \mathbf{A}]} = \frac{N^2 PE(k_i; k_i, \theta_1)\binom{|A_i|}{k_i}}{(N+1)^2 PE(k_i; |A_i|, \theta_1)}$$

Case 4.3: If $|A_i|$ points are moved from cluster A_i to existing cluster A_j, now we have one less cluster so that,

$$P(A_j, A_i : |A_i|) = (N-1)^{-2} PE(|A_i|; |A_j| + |A_i|, \theta_1) \binom{|A_i| + |A_j|}{|A_i|}^{-1}$$

The ϕ values are: $\phi[\lambda, \mathbf{A}'|\lambda, \mathbf{A}] = \beta\beta' P(A_i \xrightarrow{|A_i|} A_j)$ and $\phi[\lambda, \mathbf{A}|\lambda, \mathbf{A}'] = \beta\beta' P(A_j \xrightarrow{|A_i|} A_i)$. Together, this gives us the ratio:

$$\frac{\phi[\lambda, \mathbf{A}|\lambda, \mathbf{A}']}{\phi[\lambda, \mathbf{A}'|\lambda, \mathbf{A}]} = \frac{N^2 PE(|A_i|; |A_j| + |A_i|, \theta_1)}{(N-1)^2 PE(|A_i|; |A_i|, \theta_1)\binom{|A_i|+|A_j|}{|A_i|}}$$

Case 4.4: If $|A_i|$ points are moved from cluster A_i to a new cluster $A_{|A|+1}$:

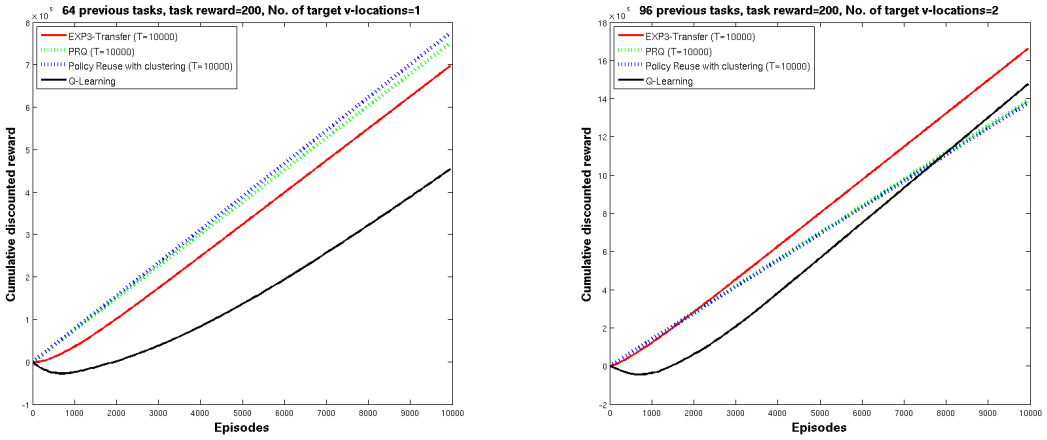
$$P(A_{N+1}, A_i : |A_i|) = N^{-2} PE(|A_i|; |A_i|, \theta_1)$$

The clustering $\mathbf{A}$ does not change in this case and the ϕ values are: $\phi[\lambda, \mathbf{A}'|\lambda, \mathbf{A}] = \beta\beta' P(A_i, A_{N+1}; |A_i|)$, $\phi[\lambda, \mathbf{A}|\lambda, \mathbf{A}'] = \beta\beta' P(A_{N+1}, A_i; |A_i|)$, which gives us

$$\frac{\phi[\lambda, \mathbf{A}|\lambda, \mathbf{A}']}{\phi[\lambda, \mathbf{A}'|\lambda, \mathbf{A}]} = 1 \tag{38}$$

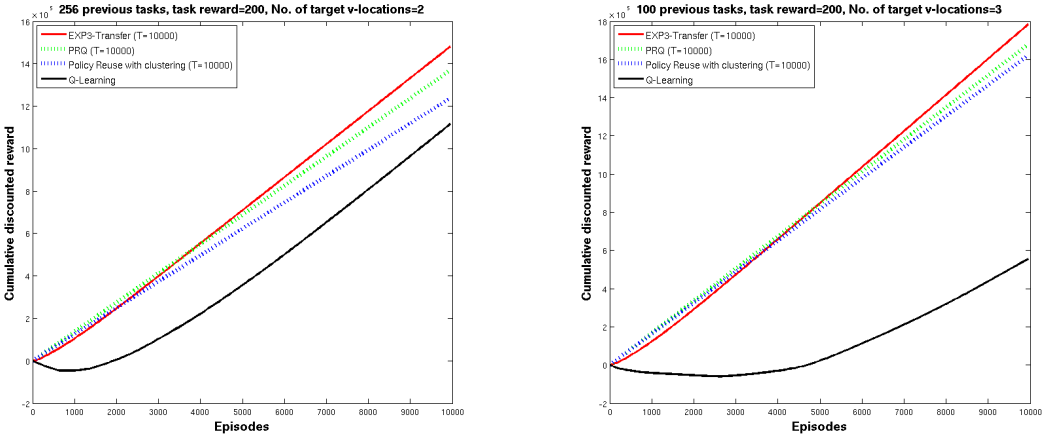
Appendix D. Surveillance Domain Experiments: Algorithm Comparisons

In this section we give detailed cumulative reward curves for the 4 algorithms: E3T with clustering, Policy-Reuse with clustering and Policy-Reuse with clustering. The results are given in Figures 16 to 18. The results more or less show what the summary graphs showed. In particular, when the number of previous tasks and the complexity of task is low, Policy-Reuse is better than our algorithm. However, as the complexity keeps increasing, our algorithm begins to dominate both versions of Policy-Reuse, showing that clustering is beneficial.



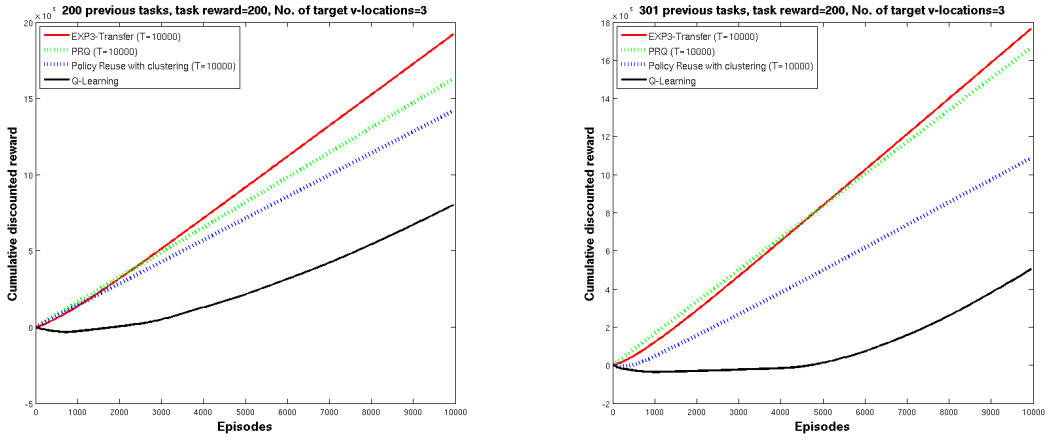
(a) (b)

Figure 16: ALGORITHM COMPARISONS. These figures compares the performance of EXP-3-Transfer with clustering, Policy-Reuse with and without clustering, and Q-learning for various settings of the task (see the figure title). These are the detailed plots of the summary results presented in Section 7.3.1.



(a) (b)

Figure 17: ALGORITHM COMPARISONS CONTINUED. These figures compares the performance of EXP-3-Transfer with clustering, Policy-Reuse with and without clustering, and Q-learning for various settings of the task (see the figure title). These are the detailed plots of the summary results presented in Section 7.3.1.

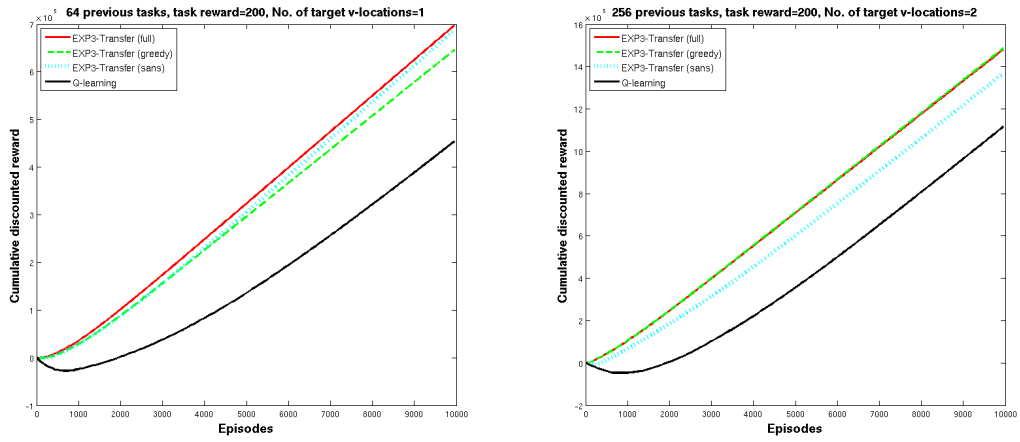


(a) (b)

Figure 18: ALGORITHM COMPARISONS CONTINUED. These figures compares the performance of EXP-3-Transfer with clustering, Policy-Reuse with and without clustering, and Q-learning for various settings of the task (see the figure title). These are the detailed plots of the summary results presented in Section 7.3.1.

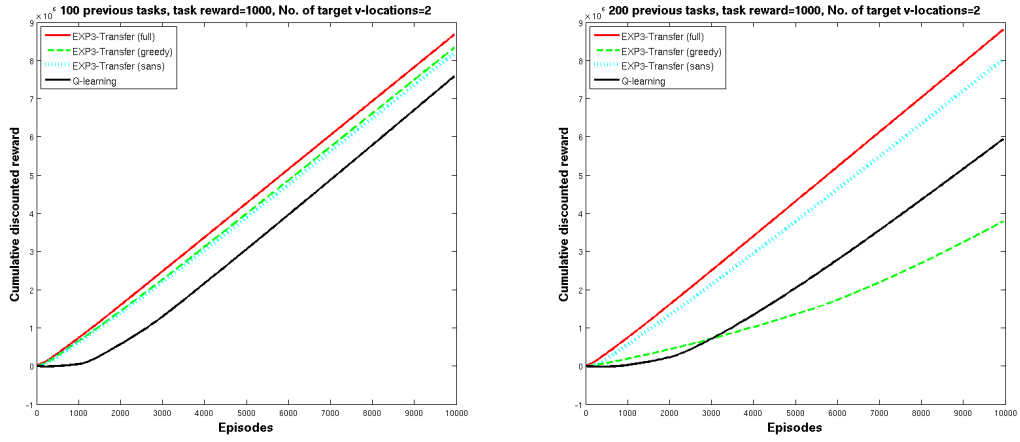
D.1 Surveillance Domain: Clustering Comparisons

In this section in figures 19 to 21 we present the learning curves summarized in figure 13. The general trend follows what was observed in Section 7.3.2.



(a) (b)

Figure 19: CLUSTERING COMPARISIONS EXTENDED RESULTS. These figures show the results that are summarized in Figure 13. The title of the graphs describe the experiment setup.



(a) (b)

Figure 20: CLUSTERING COMPARISIONS EXTENDED RESULTS CONTINUED. These figures show the results that are summarized in Figure 13. The title of the graphs describe the experiment setup.

D.2 Time Comparisons

Figures 22 to 24 gives the time comparison results for transfer problems not described in Figure 15.

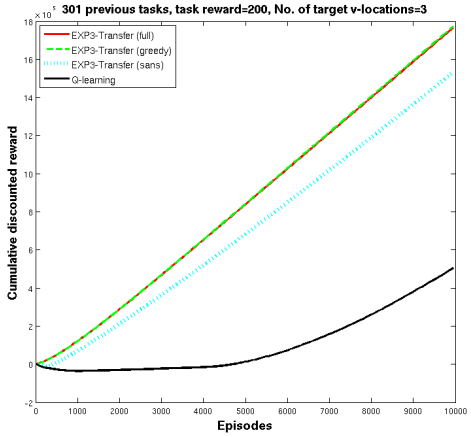
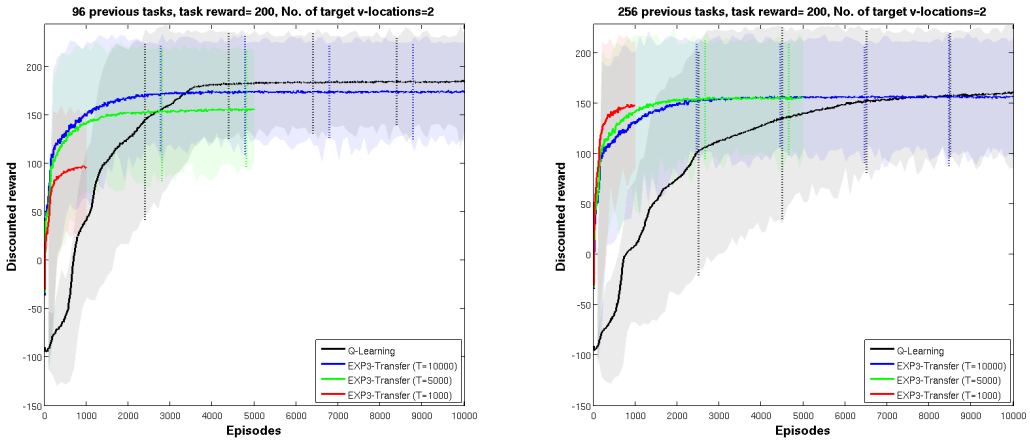
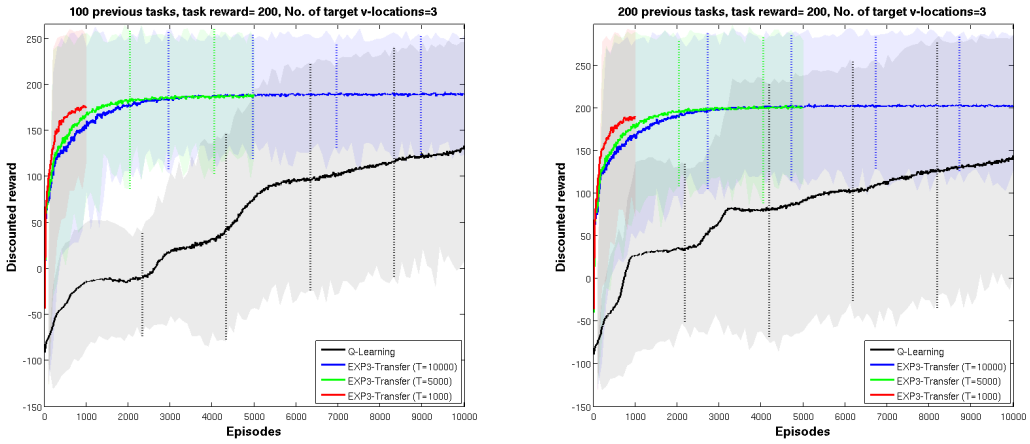


Figure 21: CLUSTERING COMPARISIONS EXTENDED RESULTS CONTINUED. These figures show the results that are summarized in Figure 13. The title of the graphs describe the experiment setup.



(a) (b)

Figure 22: TIME COMPARISIONS EXTENDED RESULTS. These figures show time comparsion results for transfer tasks in addition to Figure 15. The title of the graphs show the experiment setup. The shaded areas give the standard deviation for the learning curves.



(a) (b)

Figure 23: TIME COMPARISIONS EXTENDED CONTINUED. These figures show time comparsion results for transfer tasks in addition to Figure 15. The title of the graphs show the experiment setup. The shaded areas give the standard deviation for the learning curves.

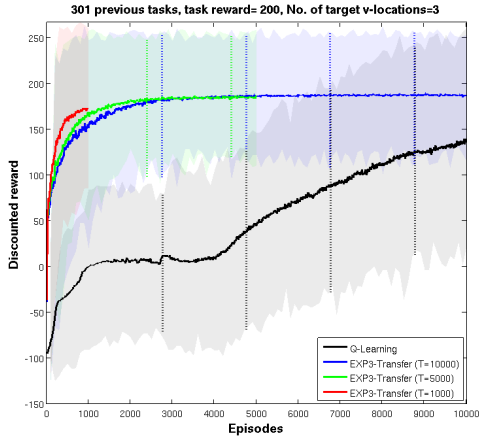


Figure 24: TIME COMPARISIONS EXTENDED CONTINUED. These figures show time comparsion results for transfer tasks in addition to Figure 15. The title of the graphs show the experiment setup. The shaded areas give the standard deviation for the learning curves.

References

Haitham Bou Ammar, Karl Tuyls, Matthew E. Taylor, Kurt Driessen, and Gerhard Weiss. Reinforcement learning transfer via sparse coding. In *Proceedings of International Conference on Autonomous Agents and Multiagent Systems*, 2012.

Bo An, David Kempe, Christopher Kiekintveld, Eric Shieh, Satinder Singh, Milind Tambe, and Yevgeniy Vorobeychik. Security games with limited surveillance. In *Proceedings of the 26th Conference on Artificial Intelligence*, 2012.

Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47:235–256, 2002a.

Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multi-armed bandit problem. *SIAM Journal on Computing*, 32:48–77, 2002b.

Mohammad Gheshlaghi Azar, Alessandro Lazaric, and Emma Brunskill. Regret bounds for reinforcement learning with policy advice. In *Proceedings of* 23^{rd} *European Conference on Machine learning (ECML)*, 2013.

Jonathan Baxter. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12:149–198, March 2000.

James L. Carroll and Kevin Seppi. Task similarity measures for transfer in reinforcement learning task libraries. In *Proceedings of 2005 IEEE International Joint Conference on Neural Networks*, 2005.

Pablo Samuel Castro and Doina Precup. Using bisimulation for policy transfer in mdps. In *Proceedings of the the* 24^{th} *AAAI Conference on Artificial Intelligence*, 2010.

Nicolo Cesa-Bianchi and Gabor Lugosi. *Prediction Learning and Games*. Cambridge University Press, 2006.

David Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554, 2003.

David Maxwell Chickering and Craig Boutilier. Learning equivalence classes of bayesian-network structures. *Journal of Machine Learning Research*, 2:150–157, 2002.

Devdatt P. Dubhashi and Alessandro Panconesi. *Concentration of Measure Inequalities for the Analysis of Randomized Algorithms*. Cambridge University Press, 2009.

Fernando Fernandez, Javier Garcia, and Manuela Veloso. Probabilistic policy reuse in a reinforcement learning agent. In *Proceedings of the 5th International Conference on Autonomous Agents and Multiagent Systems*, 2006.

Fernando Fernandez, Javier Garcia, and Manuela Veloso. Probabilistic policy reuse for inter-task transfer learning. *Robotics and Autonomous Systems*, 58:866–871, 2010.

Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for finite markov decision processes. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2004.

Eliseo Ferrante, Alessandro Lazaric, and Marcello Restelli. Transfer of task representation in reinforcement learning using policy-based protovalue functions. In *Proceedings of the* 7^th *International Confernce on Autonomous Agent And Multiagent Systems*, 2008.

John R. Hauser, Glen L. Urban, Guilherme Liberali, , and Michael Braun. Website morphing. *Marketing Science*, 28(2):202–223, March 2009.

David Heckerman and David M. Chickering. Learning bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, pages 20–197, 1995.

Richard Karp. Reducibility among combinatorial problems. In *Proceedings of a Symposium on the Complexity of Computer Computations*, 1972.

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220: 671–680, 1983.

Andrey Kolmogorov and Sergei V. Fomin. *Introductory Real Analysis*. Dover Publications, 1970.

George Konidaris and Andrew G. Barto. Building portable options: skill transfer in reinforcement learning. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, 2007.

Allesandro Lazaric and Marcello Restilli. Transferring from multiple mdps. In *Proceedings of the Neural Information Processing Systems Conference*, 2011.

Allesandro Lazaric, Marcello Restelli, and Andrea Bonarini. Transfer of samples in batch reinforcement learning. In *Proceedings of the* 25^{th} *International Conference on Machine Learning*, 2008.

David A. Levin, Yuval Peres, and Elizabeth L. Wilmer. *Markov Chains and Mixing Times*. American Mathematical Society, 2009.

Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, 1994.

M. Locatelli. Simulated annealing, algorithms for continuous global optimization: convergence conditions. *Journal of Optimization Theory and Applications*, 104(1):121–133, 2000.

Sridhar Mahadevan. Proto-value functions: Developmental reinforcement learning. In *Proceedings of International Conference on Machine Learning*, 2005.

Tom M. Mitchell and Sebastian Thrun. Explanation-based neural network learning for robot control. In *Adavances in Neural Information Processing Systems*, pages 287–294, San Mateo, CA, 1993. Morgan Kaufmann Press.

Arnab Nilim and Laurent El Ghaoui. Robust control of markov decision processes with uncertain transition matrices. *Operations Research*, 53(5):780–798, 2005.

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley and Sons, 1994.

Balaraman Ravindran. Relativized hierarchical decomposition of markov decision processes. *Decision making: neural and behavioural approaches*, 42:465–488, 2013.

Balaraman Ravindran and Andrew G. Barto. SMDP homomorphisms: An algebraic approach to astraction in semi-markov decision processes. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2003.

Christian P. Robert and George Casella. *Monte Carlo Statistical Methods*. Springer, Berling, 2005.

Jonathan Sorg and Satinder Singh. Transfer via soft homomorphisms. In *Proceedings of the* 8^th *International Conference on Autonomous Agents and Multiagent Systems*, 2009.

Alexander L. Strehl and Michael L. Littman. An analysis of model-based interval estimation for markov decision processes. *Journal of Computer Systems Science*, 74(8):1309–1331, 2008.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

Erik Talvitie and Satinder Singh. An experts algorithm for transfer learning. In *Proceedings of the* 20^{th} *International Joint Conference on Artifical Intelligence*, 2007.

Matthew Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10:1633–1685, 2009.

Sebastian Thrun. *Explanation-Based Neural Network Learning: A Lifelong Learning Approach*. Kluwer Academic Publishers, Boston, MA, 1996.

Sebastian Thrun and Tom Mitchell. Lifelong robot learning. *Robotics and Autonomous Systems*, 15:25–46, 1995.

Vladimir Vovk. Aggregating strategies. In *Proceedings of the* 3^{rd} *Internation Conference on Computational Learning Theory*, 1990.

Aaron Wilson, Alan Fern, Soumya Ray, and Prasad Tadepalli. Multi-task reinforcement learning: A hierarchical bayesian approach. In *Proceedings of the* 24^{th} *International Conference on Learning Theory*, 2007.

Jia Yuan Yu and Shie Mannor. Arbitrarily modulated markov decision processes. In *Proceedings of the IEEE Conference on Decision and Control*, 2009.

Jia Yuan Yu, Shie Mannor, and Nahum Shumkin. Markov decision processes with arbitrary reward processes. *Mathematics of Operations Research*, 34(3):737–757, 2009.